MASTER'S THESIS

Interuniversity Master in Statistics and Operations Research UPC-UB

Title: Lighting the black box: explaining individual predictions of machine learning algorithms

Author: Pol Ferrando Hernández

Advisor: Pedro Delicado Useros, Lluís Belanche Muñoz

Department: Estadística i Investigació Operativa

University: UPC-UB

Academic year: 2017-2018

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

UNIVERSITAT DE BARCELONA

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Master Thesis (MESIO)

Lighting the black box: explaining individual predictions of machine learning algorithms

Pol Ferrando Hernández

Advisors: Pedro Delicado Useros, Lluís Belanche Muñoz

Departament d'Estadística i Investigació Operativa

First and foremost, I would like to express my sincerest gratitude to my advisors, Dr. Pedro Delicado Useros and Dr. Lluís Belanche Muñoz, for the continuous support. Their guidance helped me in all the time of research and writing of this thesis. Finally, I also dedicate it to my family, that always believe in me and made me who I am today.

Abstract

Keywords: machine learning, interpretability

MSC2000: 200068T01

Many machine learning techniques remain "black boxes" because, despite their high predictive performance, it is difficult to understand the role of each variable involved in the prediction task and how it combines with others to produce a prediction. In this thesis, we have studied in depth and unified the notation of three methods found in the literature that can explain individual predictions of any model: Local Interpretable Model-agnostic Explanations (LIME), explanation vectors and Interactions-based Method for Explanation (IME). These methods determine what explanatory variables are most influential for each particular observation, which provides understanding of the reasons behind the model's decision for a particular instance and brings transparency to machine learning algorithms. In addition, we have applied these methods to an artifficial data set for which we know the role of each feature beforehand in order to analyze the strengths and weaknesses of each one of the explanation methods, compare their explanations for the same predictions and determine if the analyzed explanation methods really provide useful explanations of the reasons behind a model's decisions. Finally, we have explored their usefulness in comparing predictions of different models.

Contents

Introduction

Despite their high predictive performance, many machine learning techniques remain *black boxes* because it is difficult to understand the role of each feature and how it combines with others to produce a prediction. However, users need to understand and trust the decisions made by machine learning models, especially in sensitive fields such as medicine. For this reason, there is an increasing need of methods able to *explain* the individual predictions of a model, that is, a way to understand what features made the model give its prediction for a specific data point. Furthermore, this instance-based point of view allows detecting local peculiarities that could be unperceived in a global view (for example, due to cancellation effects).

A popular approach to this explanation problem is to use an *interpretable* (or *transparent*) model, from which an explanation can be extracted by observing its components. For instance, a decision tree can be explained by observing the rules that lead from the root to the leaves, or a generalized linear model can be explained by each feature's estimated coefficient. Interpretable models are a valid solution (and will provide meaningful insights) as long as they are accurate for the task, but limiting ourselves to this kind of models is too restrictive and it can compromise accuracy, so it is preferrable to be able to use models as flexible as needed by the problem, without restrictions.

For less transparent models (i.e, black boxes), an alternative approach is to use *model-specific* (or *model-dependent*) explanations, that is, explanation methods designed specifically for a certain type of model. For example, according to [**6**], Breiman provided additional tools for explaining the decisions of random forests, and a lot of work has been done on explaining the decisions of artificial neural networks. The problem of these methods is that the explanation method has to be replaced every time that the model is replaced with one of a different type and, consequently, the final user needs additional time and effort to get used to the new explanation method.

Therefore, we need *model-independent* (or *model-agnostic*) explanations or, in other words, a method that can explain the predictions of **any** model. With such explanations, we are not restricted to a specific type of model. Also, since such method uses the same techniques and representations to explain predictions of any model, it is easier to compare two candidate models for the task or to switch models if they are from different types.

The challenges for model-independent explanations are:

- Type of task. Supervised learning includes classification and regression. The question is: can an explanation method work for both type of tasks? That is, in addition to be model-independent, it would be nice to have a "task-independent" method or, at least, two versions of the method which share its essence.
- Interpretability. Even if a coefficient for each feature can be inspected, explanations can be not interpretable. For example, even a linear model can be difficult to interpret if it has hundreds or thousands of coefficients. The same happens if the input variables for the model are difficult to understand because explanations in terms of these variables will not be interpretable by a human and, consequently, they will be useless.

- Locally faithful. If the explanation is generated by approximating the original black box model, we have to ensure that the approximated model behaves similarly to how the original model does in the vicinity of the instance being predicted to properly capture the relationship between the input variables and the response.
- Efficiency. Users often need to understand more than a single prediction, specially if their goal is to decide trusting a model or not. Therefore, the explanation method cannot be too computationally expensive.
- Global understanding. It would be very interesting if individual predictions (local view) can help to trust a model as a whole (global view).
- Actionability. If we can obtain a global understanding of a model with explanations, can these explanations be useful to improve the original model (by feature engineering or feature selection, for instance)?

This thesis has been developed with several objectives. First, we want to introduce the model-independent methods proposed in the literature that are able to explain individual predictions of machine learning models. Second, this work wants to unify the notation of these methods to make the comparison between them easier. Third, we want to apply these methods to a data set in order to analyze the strengths and weaknesses of each one of the chosen methods and, also, compare their explanations for the same predictions. Finally, the main goal of this project is to determine if the analyzed explanation methods really provide useful explanations of the reasons behind a model's decisions.

The remaining part of this work is organized as follows. There are two parts: part 1 develops the theoretical framework of model-independent explanation methods and part 2 covers the experimental part. In chapter 1, we introduce basic concepts and notation that we need to describe the three considered explanation methods in chapters 2, 3 and 4. In chapter 5 we apply the explanation methods to a simple dataset for which we have previous knowledge of the role of each explanatory variable. In the end, the conclusions of the thesis are described and future work that would complement this study are discussed.

To finish this introduction, I would want to explain my personal motivation to choose this subject as a master thesis. I am interested in both machine learning and data mining, but there are some machine learning techniques that are opaque and it is difficult to extract meaningful insights from them. Consequently, I chose this topic to learn more about methods that bring transparency to machine learning models.

Part 1

Explanation methods

Chapter 1
Preliminaries

Data are essential in statistical modeling and machine learning. Specifically, the goal of supervised learning is to predict the value of an outcome measure based on a number of input measures. These input measures that we observe are called *features*[1], and the space where they are placed is called the *feature space*.

Definition 0.1. Feature space
The feature space $\mathcal{X}$ is the cartesian product of p features.

$$\mathcal{X} = X_1 \times X_2 \times ... \times X_p$$

We will often consider $\mathcal{X} = \mathbb{R}^p$.

Supervised learning algorithms try to find a useful approximation (a *model*) to the underlying function that determines the relationship between the features and the responses, in the sense that the model is able to properly predict the outcome for unseen instances. To do so, methods learn from a set of examples for which both features and outcome have been measured, known as training data.

Furthermore, we distinguish two types of predicting tasks depending on the type of the output: *classification* tries to predict categorical outputs, while *regression* attempts to predict quantitative outputs.

Definition 0.2. Classification model (or classifier)
Let $\{1, ..., C\}$ be a finite set of class labels. A classifier $f = (f_1, f_2, ..., f_C)$ is a mapping from a feature space to a C-dimensional space

$$f : \mathcal{X} \to [0, 1]^C$$

such that $\sum_{c=1}^{C} f_c(x) = 1$, $\forall x \in \mathcal{X}$.

In classification, there are two types of classifiers: *hard* classifiers, which directly predict the class label of observations, and *probabilistic* (or *soft* classifiers), whose outputs can be interpreted as probabilities. Note that Definition 0.2 includes hard classifiers when we consider that the component of the predicted class is 1 and the rest are 0.

Definition 0.3. Regression model
A regression model f is a mapping from a feature space to the real numbers

$$f : \mathcal{X} \to \mathbb{R}$$

In addition to predictive accuracy, which is not the focus of this work, an important part of data analysis is to gain insights into the underlying function that determines the relationship between the

[1]Also known as *inputs*, *predictors*, *dependent variables* or *explanatory variables*, specially in statistical literature.

features and the responses. As discussed in the introduction (see Section), there are models inherently interpretable whose components can directly provide this understanding. However, many machine learning techniques (e.g., support vector machines or artifficial neural networks) are less transparent in this aspect and it is not clear what features are more influential on the model's decision, which is the reason why they are called *black boxes*.

For these black box models, explanation methods try to provide qualitative understanding on what input features made a model give its prediction for a particular instance. Therefore, explanation methods assume that the user already has access to a trained model f and one observation $\tilde{x} \in \mathcal{X}$ whose prediction wants to be justified. Furthermore, this work focuses on model-independent methods, which do not make any assumption about f, so our goal is to produce an explanation for a prediction only using the predictions made by the model.

Finally, the explanation methods discussed in this work will highlight the most influential features on the model's prediction and show them with visual artifacts to facilitate interpretability. However, it is important to keep in mind that their goal is to explain the prediction made by an specific model, so the generated explanation will be determined by the model. Consequently, on the one hand, these methods will provide meaningful insights if the model has really captured the relationship between the predictors and the outcome. On the other hand, if the model is inaccurate, explanation methods will make the user notice that the model prediction is not reliable and that the model needs to be checked in order to be trustworthy.

Chapter 2
Local Interpretable Model-agnostic Explanations (LIME)

1. Intuition behind LIME

LIME (see Ref. [**1**] and [**2**]) stands for *Local Interpretable Model-agnostic Explanations*, a name that perfectly summarizes the three basic ideas behind this explanation method: *model-agnosticism*, *interpretability* and *locality*.

First, LIME do not make any assumptions about the black-box model because its purpose is to be *model-independent* (or model-agnostic, as Ribeiro et al. say). Consequently, the only way that LIME has to understand the black-box's behavior is perturbing the input and see how the predictions change.

Second, Ribeiro et al. argue that, for the benefit of *interpretability*, the input variables used to produce explanations may need to be different from the actual features used by the model. According to them, explanations have to provide qualitative understanding of the relationship between the input variables and the response, but they have to be easy to understand by users above all, which is not necessarily true of the feature space used by the model as we have seen in section **??**. For this reason, LIME's explanations use a data representation (called *interpretable representation*) that is different from the original feature space.

Finally, according to LIME's authors, it is much easier to approximate a black-box model by a simple model locally (i.e., in the neighborhood of the instance we want to explain) rather than approximate it globally. For this reason, LIME produces an explanation by approximating the black-box model by an *interpretable* model (for instance, a linear model with a few non-zero coefficients) learned on perturbations of the original instance weighted by their similarity to it.

In summary, LIME generates an explanation for a prediction from the components of an interpretable model (for instance, the coefficients in a linear regression) which resembles the black-box model at the vicinity of the point of interest and which is trained over a new data representation to ensure interpretability.

2. Basic Concepts

2.1. Interpretable representation. To ensure that the explanation is *interpretable*, Ribeiro et al. (see Ref. [**1**] and [**2**]) distinguish an *interpretable representation* $\mathcal{X}'$ from the *original* feature space $\mathcal{X}$ that the model uses.

The interpretable representation has to be understandable to humans, so its dimension is not necessarily the same as the dimension of the original feature space. Let p be the dimension of the original feature space $\mathcal{X}$ and let p' be the dimension of the interpretable space $\mathcal{X}'$.

The interpretable inputs map to the original inputs through a mapping function $h_{\tilde{x}} : \mathcal{X}' \to \mathcal{X}$ (see Reference [**?**]), specific to the instance $\tilde{x}$ we want to explain. Different types of mappings are used for different input spaces.

For text data, a possible interpretable representation is a binary vector indicating the presence or absence of a word, although the classifier may use more complex and incomprehensible features such as word embeddings. Formally, $\mathcal{X}' = \{0,1\}^{p'} \equiv \underbrace{\{0,1\} \times \cdots \times \{0,1\}}_{\text{(p')}}$ where p' is the number of words that contains the instance being explained $\tilde{x}$ and the mapping function converts a vector of 1's or 0's (presence or absence of a word, respectively) into the representations used by the model: if it uses word counts, the mapping $h_{\tilde{x}}$ will map 1 to the original word count and 0 to 0; but if the model uses word embeddings, the mapping should convert any sentence expressed as a vector of 1's and 0's into its embedded version.

For images, a possible interpretable representation is a binary vector indicating the presence or absence of a set of contiguous similar pixels (also called super pixel). Formally, $\mathcal{X}' = \{0,1\}^{p'}$ where p' is the number of super pixels considered, usually obtained by an image segmentation algorithm such as quick shift (see Figure 1 for an example). In this case, the mapping function maps 1 to leaving the super pixel as in the original image and 0 to gray the super pixel out (which represents being missing).

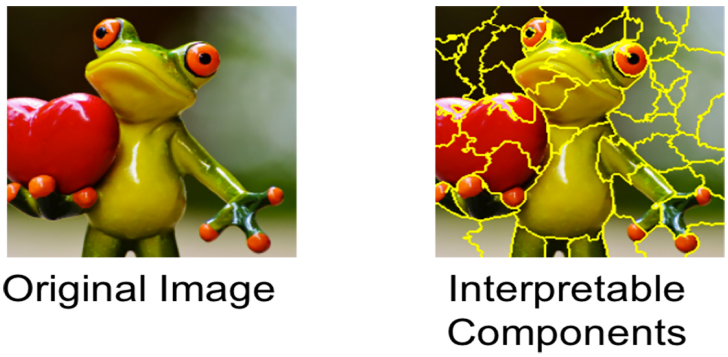


FIG. 1. Original representation (left) and interpretable representation (right) of an image. Sets of contiguous similar pixels (delimited by yellow lines) are called super pixels. Each super pixel defines one interpretable feature. Source: https://www.oreilly.com/learning/introduction-to-local-interpretable-model-agnostic-explanations-lime

For tabular data (i.e., matrices), the interpretable representation depends on the type of features: categorical, numerical or mixed data. For categorical data, $\mathcal{X}' = \{0,1\}^p$ where p is the actual number of features used by the model (i.e., $p' = p$) and the mapping function maps 1 to the original class of the instance and 0 to a different one sampled according to the distribution of training data. For numerical data, $\mathcal{X}' = \mathcal{X}$ and the mapping function is the identity. However, we can discretize numerical features so that they can be considered categorical features. For mixed data, the interpretable representation will be a p-dimensional space composed of both binary features corresponding to categorical features and numerical features corresponding to numerical features (if they are not discretized). Each component of the mapping function is also defined according to the previous definitions depending on the type of the explanatory variable.

2.2. Locality. Although a simple model may not be able to approximate the black box model globally, approximating it in the neighborhood of the individual instance we want to explain may be feasible. In other words, LIME relies on the assumption that every complex model is linear on a local scale.

Formally, we need a weight function $w_{\tilde{x}} : \mathcal{X} \to \mathbb{R}^+$ that gives the most weight to the instances $z \in \mathcal{X}$ closest to the instance we want to explain $\tilde{x} \in \mathcal{X}$ and the least weight to the instances that are furthest away. Ribeiro et al. (see Ref. [**1**] and [**2**]), this role as a similarity measure between any instance to $\tilde{x}$ is said to define *locality* around the instance $\tilde{x}$.

2.3. Fidelity-interpretability trade-off. Ribeiro et al. (see Ref. [**1**] and [**2**]) define an explanation as a model $g : \mathcal{X}' \to \mathbb{R}$ such that $g \in G$, where G is a class of "potentially" interpretable models. Note that the domain of g is the interpretable space $\mathcal{X}'$. According to them, "potentially" interpretable models are models that can be readily presented to the user with visual or textual artifacts, such as linear models or decision trees.

Let $\mathcal{L}(f, g, w_{\tilde{x}})$ be a *loss function* that measures how accurate g is in approximating f taking into account the weights $w_{\tilde{x}}$. As not every $g \in G$ may be simple enough to be interpretable, we let $\Omega(g)$ be a regularization term that measures the complexity (as opposed to interpretability) of the explanation $g \in G$.[1] For example, for linear models $\Omega(g)$ may be the number of non-zero coefficients, while for decision trees $\Omega(g)$ may be the depth of the tree.

3. Method

Let $\mathcal{X} = \mathbb{R}^p$ be the feature space. Let $\tilde{x} \in \mathbb{R}^p$ be the *original representation* of an instance being explained, and $\tilde{x}' \in \mathcal{X}'$ be its *interpretable representation.*

Also, let $f : \mathcal{X} = \mathbb{R}^p \to \mathbb{R}$ be the model being explained. In classification, $f(x)$ is the probability (or a binary indicator) that x belongs to a certain class. For multiple classes, LIME explains each class separately, thus $f(x)$ is the prediction of the relevant class. In regression, $f(x)$ is the regression function.

Finally, let $g : \mathcal{X}' \to \mathbb{R}$ be the explanation model. Let $\mathcal{L}(f, g, w_{\tilde{x}})$ be a *loss function* that measures how unfaithful g is in approximating f in the locality defined by $w_{\tilde{x}}$, and let $\Omega(g)$ be a measure of complexity of the explanation $g \in G$.

In order to ensure both interpretability and local fidelity, we must minimize $\mathcal{L}(f, g, w_{\tilde{x}})$ while having $\Omega(g)$ be low enough to be interpretable by humans. Thus, the explanation produced by LIME is obtained by the following:

$$\xi(\tilde{x}) = \underset{g \in G}{\operatorname{argmin}} \Big\{ \mathcal{L}(f, g, w_{\tilde{x}}) + \Omega(g) \Big\} \tag{3.1}$$

Note that this formulation can be used with different explanation families G, loss functions $\mathcal{L}$ and regularization terms Ω.

In practice, the general approach LIME uses to produce an explanation is the following:

(1) Generate N "perturbed" samples of the interpretable version of the instance to explain $\tilde{x}'$. Let $\{z'_i \in \mathcal{X}' \mid i = 1, ..., N\}$ be the set of these observations.
(2) Recover the "perturbed" observations in the original feature space by means of the mapping function. Let $\big\{z_i \equiv h_{\tilde{x}}(z'_i) \in \mathcal{X} \mid i = 1, ..., N\big\}$ be the set in the original representation.

[1] References [**1**] and [**2**] refer to $\mathcal{L}(f, g, w_{\tilde{x}})$ as *fidelity function* and to $\Omega(g)$ as *complexity measure.*

(3) Let the black box model predict the outcome of every "perturbed" observation. Let $\left\{ f(z_i) \in \mathbb{R} \mid i = 1, ..., N \right\}$ be the set of responses.
(4) Compute the weight of every "perturbed" observation. Let $\{ w_{\tilde{x}}(z_i) \in \mathbb{R}^+ \mid i = 1, ..., N \}$ be the set of weights.
(5) Solve the optimization problem (3.1) using the dataset of "perturbed" samples with their responses $\mathcal{Z} = \left\{ \left(z'_i, f(z_i) \right) \in \mathcal{X}' \times \mathbb{R} \mid i = 1, ..., N \right\}$ as training data.

Note that the complexity does not depend on the size of the training set (because it produces the explanation for an individual prediction), but instead on time to compute $f(z)$ and on the number of samples N.

The sampling process described in step 1 depends on the interpretable space $\mathcal{X}'$, so it is different depending on the type of input data (see Section 2.1). For text data or images, whose interpretable space is composed of binary features (i.e., $\mathcal{X}' = \{0, 1\}^{p'}$), the samples $z'_i \in \mathcal{X}'$ are obtained by drawing non-zero elements of $\tilde{x}'$ uniformly at random, where the number of such draws is also uniformly sampled. For tabular data (i.e., matrices), step 1 depends on the type of features and it requires the original training set.

For text data, since the interpretable space is a binary vector indicating the presence or absence of a word, the process means randomly removing words from the instance being explained. For example, if we are trying to explain the prediction of a text classifier for the sentence "I hate this movie", we will perturb the sentence and get predictions on sentences such as "I hate movie", "I this movie", "I movie", "I hate", etc. Note that if the classifier uses some uninterpretable representation such as word embeddings, this still works: in step 2 we will represent the perturbed sentences with word embeddings, and the explanation will still be in terms of words such as "hate" or "movie".

For images, since the interpretable space is a binary vector indicating the presence or absence of a super pixel, the process consists of randomly making some of the super pixels gray. In other words, if we want to explain the prediction for an image, we will perturb the image and get predictions on images which have one or more hidden super pixels. This process is described in Figure 2.

For tabular data (i.e., matrices), there are differences between categorical and numerical features, but both types are dependent on the training set. For categorical features (whose interpretable representation is binary) perturbed samples are obtained by sampling according to the training distribution and making a binary feature that is 1 when the value is the same as the instance being explained. For numerical features, the instance being explained is perturbed by sampling from a Normal(0,1) distribution and doing the inverse operation of mean-centering and scaling (using the means and standard deviations of the training data).

4. Sparse Linear Explanations

Ribero et al. (see Ref. [**1**]) focus on what they call *sparse linear explanations*, which correspond to specific choices of the explanation familiy G, the loss function $\mathcal{L}$ and the regularization term Ω. The choices are not explicitly justified, but some probable reasons can be inferred. They are the following:

- The class of linear models as explanation family G.

$$g(z') = \beta \cdot z'$$

  Linear models let us measure each feature's influence in a prediction by inspecting both the magnitude and the sign of its coefficient. Therefore, linear models provide qualitative understanding between the explanatory variables and the response, which makes them potentially interpretable models suitable for being explanation models.

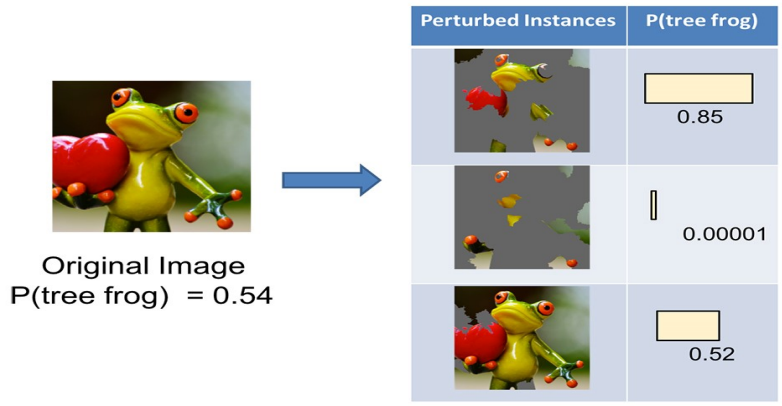


FIG. 2. Examples of perturbed instances of an image and their predictions. Source: https://www.oreilly.com/learning/introduction-to-local-interpretable-model-agnostic-explanations-lime

- An exponential kernel defined on some distance function D as weight function.

$$w_{\tilde{x}}(z) = e^{-D(\tilde{x},z)^2/\sigma^2}$$

The choice of using an exponential kernel is not justified in Ref. [**1**] nor [**2**]. The distance functions used are the cosine similarity measure[2] for text data (which is the standard in text analysis according to Ref. [**11**]) and the Euclidean distance[3] for images and tabular data. However, the implementations of LIME (see Ref. [**10**] and [**11**]) accept any other distance chosen by the user. Finally, even though the choice of the kernel width σ is not discussed, it defaults to $\sigma = \frac{3}{4}\sqrt{p}$, where p is the number of features.

- Weighted least squares error as loss function.

$$\mathcal{L}(f, g, w_{\tilde{x}}) = \sum_{i=1}^{N} w_{\tilde{x}}(z_i)\Big(f(z_i) - g(z_i')\Big)^2$$

where $z_i' \in \mathcal{Z}$ (the dataset of "perturbed" samples with their responses described in section 3) and $z_i \equiv h_{\tilde{x}}(z_i')$.
Quadratic loss functions are often more mathematically tractable than other loss functions and, specifically, the weighted least squares problem has a closed form solution.

- Limiting to K the number of non-zero coefficients as regularization term.

$$\Omega(g) = \infty \mathbb{1}_{\{||\beta||_0 > K\}} \equiv \begin{cases} \infty \text{ , if } ||\beta||_0 > K \\ 0 \text{ , if } ||\beta||_0 \leq K \end{cases}$$

where $||\beta||_0 = \sum_{j=1}^{p'} |\beta_j|^0$ is the ℓ_0 "norm" [4] (i.e., the number of non-zero entries of β).
The regularization parameter K is crucial for the explanation task because it establishes the number of components that will be presented to the user. Ribeiro et al. (see Ref. [**1**]) say that

[2] $D(\tilde{x}, z) = \frac{\tilde{x} \cdot z}{||\tilde{x}||||z||} = \frac{\sum_{i=1}^{p} \tilde{x}_i z_i}{\sqrt{\sum_{i=1}^{p} \tilde{x}_i^2}\sqrt{\sum_{i=1}^{p} z_i^2}}$

[3] $D(\tilde{x}, z) = ||\tilde{x} - z|| = \sqrt{\sum_{i=1}^{p} (\tilde{x}_i - z_i)^2}$

[4] It is not a real norm.

K can be adapted to be as big as the user can handle but they used a constant value for K in their experiments, saying that they leave the exploration of different values to future work.

Note that there are two hyperparameters: the kernel width σ and the number of features K used for the explanation.

With the previous choices, Eq. (3.1) becomes:

$$\xi(\tilde{x}) = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} e^{-D(\tilde{x},z)^2/\sigma^2} \Big(f(z_i) - \beta \cdot z_i' \Big)^2 + \infty \mathbb{1}_{\{||\beta||_0 > K\}} \right\} \tag{4.1}$$

According to Ref. [**1**], Eq. (4.1) cannot be directly solved due to this particular choice of Ω, so they approximate the solution by first selecting K features using a feature selection technique and then estimating the coefficients via weighted least squares. In Ref. [**1**], the authors select K features using the regularization path produced by LASSO (i.e., the estimated coefficients by LASSO as a function of the shrinkage factor) and then estimate the coefficients via weighted least squares[5].

Nevertheless, implementations of LIME (see Ref. [**10**] and [**11**]) currently support two more feature selection algorithms for selecting the K features:

- Forward selection: features are added one by one based on their improvements to a ridge regression fit of the complex model outcome.
- Highest coefficients: the K features with highest absolute weight in a ridge regression fit of the complex model outcome are chosen.

Also, all features can be used for the explanation, but it is not recommended unless there are very few features.

In summary, the current approach that LIME uses to approximate the solution of equation (4.1) and produce an explanation for instance $\tilde{x}$ is:

(1) Generate N "perturbed" samples of the interpretable version of the instance to explain $\tilde{x}'$. Let $\{z_i' \in \mathcal{X}' \mid i = 1, ..., N\}$ be the set of these observations.
(2) Recover the "perturbed" observations in the original feature space by means of the mapping function. Let $\big\{z_i \equiv h_{\tilde{x}}(z_i') \in \mathcal{X} \mid i = 1, ..., N\big\}$ be the set in the original representation.
(3) Let the black box model predict the outcome of every "perturbed" observation. Let $\Big\{ f\big(z_i\big) \in \mathbb{R} \;\Big|\; i = 1, ..., N \Big\}$ be the set of responses, and let $\mathcal{Z} = \Big\{ \Big(z_i', f\big(z_i\big) \Big) \in \mathcal{X}' \times \mathbb{R} \;\Big|\; i = 1, ..., N \Big\}$ be the the dataset of "perturbed" samples with their responses.
(4) Compute the weight of every "perturbed" observation. Let $\{w_{\tilde{x}}\big(z_i\big) \in \mathbb{R}^+ \mid i = 1, ..., N\}$ be the set of weights.
(5) Select K features best describing the black box model outcome from the perturbed dataset $\mathcal{Z}$.
(6) Fit a weighted linear regression model[6] to a feature-reduced dataset composed of the K selected features in step 5. If the black box model is a regressor, the linear model will predict the output of the black box model directly. If the black box model is a classifier, the linear model will predict the probability of the chosen class.
(7) Extract the coefficients from the linear model and use them as explanations for the black box model's local behavior.

An example of this procedure for images is shown in Figure 3. Note that, in addition to a black box model (classifier or regressor) f and an instance to explain $\tilde{x}$ (and its interpretable representation

[5]Ribeiro et al. call this procedure K-LASSO.
[6]Actually, current implementations of LIME (see Ref. [**10**] and [**11**]) fit a weighted ridge regression with the regularization parameter set to 1.

$\tilde{x}'$), the previous procedure requires setting in advance the number of samples N, the kernel width σ and the length of the explanation K.

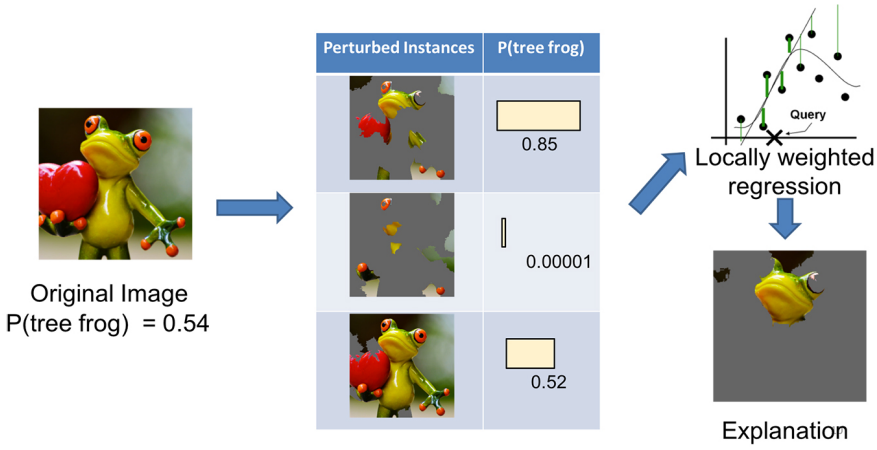


FIG. 3. Explaining the prediction of a classifier with LIME. Source: https://www.oreilly.com/learning/introduction-to-local-interpretable-model-agnostic-explanations-lime

Chapter 3
Explanation vectors

1. Intuition behind explanation vectors

According to Ref. [**4**], explaining a decision of a black box model implies understanding what input features made the model give its prediction for the observation being explained.

Intuitively, a feature has a lot of influence on the model decision if small variations in its value cause large variations of the model's output, while a feature has little influence on the prediction if big changes in that variable barely affect the model's output.

Since a model is a scalar function (see Section 1), its gradient points in the direction of the greatest rate of increase of the model's output, so it can be used as a measure of features' influence. In classification tasks, if c is the predicted class for an instance, the gradient of the conditional probability $P(Y \neq c|X = x)$ evaluated at the instance points in the direction to where the data point has to be moved to change its predicted label, so it provides qualitative understanding of the most influential features in the model decision. Similarly, although Baehrens et al. (see Ref. [**4**]) left to future work the generalization to regression tasks, it seems natural to use the gradient of the regression function because it points in the direction to where local small variations of the data point would mean large variations of the model's output.

In summary, explanations can be defined as gradient vectors that characterize how a data point has to be moved to change its prediction, which is the reason why they are called *explanation vectors*.

2. Method for classification

Baehrens et al. (see Reference [**4**]) propose to explain a prediction of any classifier by the gradient at the point of interest of the conditional probability of the class not being the predicted one given its feature values. Therefore, the explanation is a vector (called *explanation vector*) which characterizes how the data point has to be moved to change the predicted label of the instance.

This definition can be directly applied to *probabilistic* (or *soft*) classifiers because they explicitly estimate the class conditional probabilities. However, *hard* classifiers (e.g., support vector machines) directly estimate the decision rule, that is, they assign the predicted class label without producing a probability estimation. For this type of classifiers, Baehrens et al. propose to use a kernel density estimator (also called Parzen-Rosenblatt window) to estimate the class conditional probabilities so that we can approximate the hard classifier by a probabilistic classifier.

2.1. Probabilistic classifiers. Let $\mathcal{X} = \mathbb{R}^p$ be the feature space. Let $X = (X_1, ..., X_p) \in \mathbb{R}^p$ be a vector of p continuous random variable and Y be a discrete random variable with possible values (class labels) in $\{1, ..., C\}$. Let $P(X, Y)$ be the joint distribution, which is unknown most of the time.

Let $f : \mathbb{R}^p \to [0,1]^C$ such that $\sum_{c=1}^{C} f_c(x) = 1$, $\forall x \in \mathcal{X}$, be the "probabilistic" classifier being explained. Furthermore, we assume that all components of f are first-order differentiable with respect to X for all classes c and over the entire input space. Finally, let $\tilde{x} \in \mathbb{R}^p$ be the observation being explained.

Each component f_c of a prediction $f(x)$ is an estimation of the conditional probability $P(Y = c|X = x)$. However, "probabilistic" classifiers can then be turned to "hard" classifiers using the Bayes rule, which is the optimal decision rule for the 0-1 loss function:

$$\hat{y} = \underset{c\in\{1,...,C\}}{\operatorname{argmin}} \left\{1 - f_c(x)\right\}$$

Note that $\left\{1 - f_c(x)\right\}$ is an estimation of $P(Y \neq c|X = x)$. That is, the predicted class is the one which has the highest probability.

Baehrens et al. (See Ref. [**4**]) define the *explanation vector* of a data point $\tilde{x}$ as the gradient at $x = \tilde{x}$ of the conditional probability of $Y \neq \hat{y}$ given $X = x$.

Definition 2.1. Let $\{1, ..., C\}$ be a finite set of class labels and let $f : \mathbb{R}^p \to [0,1]^C$ be a classifier. Let $\hat{y}$ be the predicted class label by the classifier f. The *explanation vector* of a data point $\tilde{x}$ is:

$$\zeta(\tilde{x}) := \nabla \left(1 - f_{\hat{y}}(x)\right)\Big|_{x=\tilde{x}}$$

Note that $\left(1 - f_{\hat{y}}(x)\right)$ is an estimation of $P\big(Y \neq \hat{y}|X = x\big)$. Thus, the explanation vector $\zeta(\tilde{x})$ is a p-dimensional vector that points in the direction to where the data point has to be moved to change its predicted class. A positive (negative) sign of a component implies that increasing the corresponding feature would lower (raise) the probability that $\tilde{x}$ is assigned to $\hat{y}$. Furthermore, the larger is the absolute value of a component, the more influential is that feature in the class label prediction.

One issue with explanation vectors is that they can become a zero vector. According to Ref. [**4**], if this happens because of a local maximum or minimum, we can learn from the eigenvectors of the Hessian the features that are relevant for the model decision even though we will not obtain an orientation for them. However, if the classifier outcome is a probability distribution which is flat in some neighborhood of $\tilde{x}$, no meaningful explanation can be obtained. In summary, the explanation vectors method fits well to classifiers that outputs a probability of the class function not completely flat.

In the case of binary classification, Baehrens et al. (see Ref. [**4**]) define the *explanation vector* as the local gradient of the probability function $P(Y = 1|X = x)$ of the learned model for the positive class. Formally:

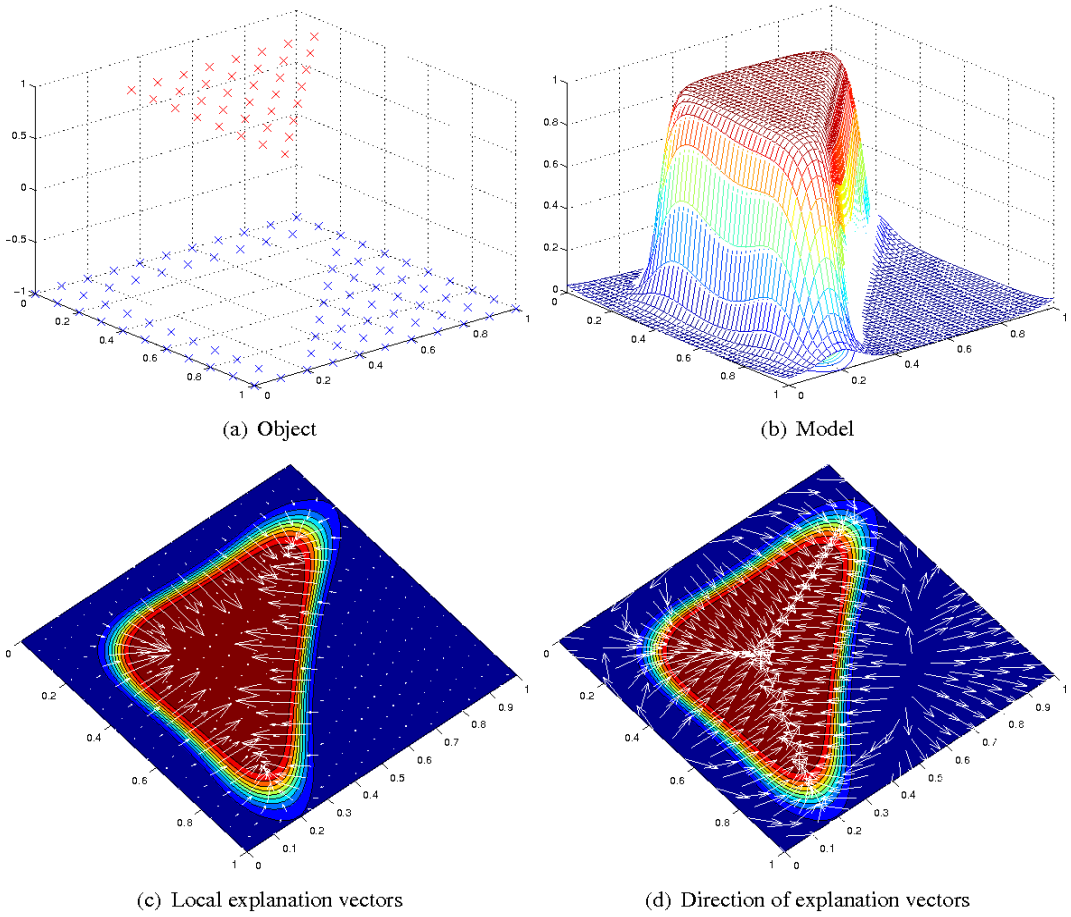
Definition 2.2. Let $f : \mathbb{R}^p \to [0,1]$ be a binary classifier. The *explanation vector* of a data point $\tilde{x}$ is:

$$\zeta(\tilde{x}) := \nabla\, f(x)\Big|_{x=\tilde{x}}$$

Note that $f(x)$ is an estimation of $P(Y = 1|X = x)$. Therefore, the explanation vector points in the direction of the steepest ascent from the data point to higher probabilities for the class 1. Also, the sign of each component indicates whether the predicted probability of being 1 would increase or decrease when the corresponding feature of $\tilde{x}$ is increased locally. Finally, note that Def. 2.2 will differ from Def. 2.1 when $\hat{y} = 1$, case in which the negative version $-\zeta(\tilde{x})$ may be especially helpful because it indicates how to move the data point to be assigned to class 0.

Fig. 1 shows an example of how explanation vectors are applied to explain predictions of a binary classifier: we have labeled training data (Fig. 1(a)) that we use to train a model (in this case, a Gaussian Process Classifier), which assigns a probability of being in the positive class to every data point of the feature space (Fig. 1(b)). Then, we compute explanation vectors of the data points we

want to explain in order to understand what features made the model give its predictions. For instance, in Fig. 1(c)-(d) we can see that explanation vectors along the hypothenuse and at the corners of the triangle produced by the data have both components different from zero, while explanation vectors along the edges only have one non-zero component. Thus, both explanatory variables influence the model decision for observations along the hypothenuse and at the corners of the triangle, while only one feature was relevant to the model prediction for observations along the edges. Furthermore, the length of the explanation vectors (Fig. 1(c)) represents the degree of importance of the relevant features, so we can compare explanations between observations.



(a) Object (b) Model

(c) Local explanation vectors (d) Direction of explanation vectors

FIG. 1. Example of how explanation vectors are applied to model predictions learned by a Gaussian Process Classifier (GPC), which provides probabilistic predictions. Panel (a) shows the training points and their labels (class +1 in red, class -1 in blue). Panel (b) shows the trained model, which assigns a probability of being in the positive class to every data point. Panels (c) and (d) show the explanation vectors and the direction of the explanation vectors, respectively, together with the contour map of the model. Source: Reference [**4**]

.

2.2. Hard classifiers. Let $\mathcal{X} = \mathbb{R}^p$ be the feature space. Let $X = (X_1, ..., X_p) \in \mathbb{R}^p$ be a vector of p continuous r.v. and Y be a discrete random variable with possible values (class labels) in $\{1, ..., C\}$. Let $P(X, Y)$ be the joint distribution, which is unknown.

Let $f : \mathcal{X} = \mathbb{R}^p \to \{1, ..., C\}$ be the "hard" classifier being explained. Finally, let $\tilde{x} \in \mathbb{R}^p$ be the observation being explained.

Let $x^1, ..., x^n \in \mathbb{R}^p$ be training points with labels $y^1, ..., y^n \in \{1, ..., C\}$. Let $I_c = \{i \in \{1, ..., N\} | f(x^i) = c\}$ be the set of indexes of the training set whose predicted labels are c. According to Ref. [**4**], for every class c, the conditional probability $P(Y = c|X = x)$ can be approximated by the following quotient of kernel density estimators:

$$\hat{f}_c(x) = \widehat{P_\sigma(Y = c|X = x)} \approx \frac{\sum\limits_{i \in I_c} k_\sigma(x - x^i)}{\sum\limits_i k_\sigma(x - x_i)}, \quad \forall c \in \{1, ..., C\} \tag{2.1}$$

where $k_\sigma(z)$ is a Gaussian kernel $\Big($that is, $k_\sigma(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z^\intercal z}{2\sigma^2}}\Big)$.

Consequently, we can define a classifier $\hat{f}_\sigma(x) = (\hat{f}_1(x), \hat{f}_2(x), ..., \hat{f}_C(x))$ whose components are estimations of $P(Y = c|X = x)$. Note that we can use the trained classifier f to generate as much labeled data as we want for constructing $\hat{f}_\sigma$. As we did for probabilistic classifiers, we can use the Bayes rule in order to get a predicted class:

$$\hat{y}_\sigma(x) = \underset{c \in \{1,...,C\}}{\operatorname{argmin}} \widehat{P_\sigma(Y \neq c|X = x)} = \underset{c \in \{1,...,C\}}{\operatorname{argmin}} \big\{1 - \hat{f}_c(x)\big\}$$

Then, Baehrens et al. (see Reference [**4**]) define the estimated *explanation vector* of a data point $\tilde{x} \in \mathbb{R}^p$ as follows:

Definition 2.3. Let $f : \mathbb{R}^p \to \{1, ..., C\}$ be a "hard" classifier. The estimated *explanation vector* of a data point $\tilde{x} \in \mathbb{R}^p$ is:

$$\hat{\zeta}(\tilde{x}) := \nabla \left. \widehat{P(Y \neq f(\tilde{x})|x)} \right|_{x=\tilde{x}} =$$

$$= \frac{\Big(\sum\limits_{i \notin I_{f(\tilde{x})}} k_\sigma(\tilde{x} - x^i)\Big)\Big(\sum\limits_{i \in I_{f(\tilde{x})}} k_\sigma(\tilde{x} - x^i)(\tilde{x} - x^i)\Big) - \Big(\sum\limits_{i \notin I_{f(\tilde{x})}} k_\sigma(\tilde{x} - x^i)(\tilde{x} - x^i)\Big)\Big(\sum\limits_{i \in I_{f(\tilde{x})}} k_\sigma(\tilde{x} - x^i)\Big)}{\sigma^2\Big(\sum_{i=1}^n k_\sigma(\tilde{x} - x^i)\Big)^2}$$

Note that $f(\tilde{x})$ is used instead of $\hat{y}_\sigma(\tilde{x})$. This is done to ensure that $\hat{\zeta}(\tilde{x})$ points in the direction to where the observation has to be moved to change the actual label predicted by f, instead of the one assigned by the approximated classifier $\hat{y}_\sigma$ (which could be different).

Finally, according to Ref. [**4**], the single hyperparameter σ is chosen such that the "new" predicted class labels $\hat{y}_\sigma$ are as similar as possible to the actual predictions made by the original "hard" classifier f on a test set. Let $z^1, ..., z^m \in \mathbb{R}^p$ be test data points, and let $f(z^1), ..., f(z^m)$ be the labels predicted by the classifier for the test points. Then, the chosen value for σ is:

$$\hat{\sigma} := \underset{\sigma}{\operatorname{argmin}} \sum_{j=1}^m \mathbb{1}_{\{f(z^j) \neq \hat{y}_\sigma(z^j)\}}$$

Chapter 4
Interactions-based Method for Explanation (IME)

1. Intuition behind IME

When a model gives a prediction for an observation, all features do not play the same role: some of them may have a lot of influence on the model's prediction, while others may be irrelevant. Consequently, one may think that the effect of each feature can be measured by checking what the prediction would have been if that feature was absent; the bigger the change in the model's output, the more important should be the feature.

However, observing only a single feature at a time implies that dependencies between features are not taken into account, which could produce inaccurate and misleading explanations of the model's decision-making process according to Ref. [**6**] and [**7**]. Therefore, to avoid missing any interaction between features, we should observe how the prediction changes for each possible subset of features and then combine these changes to form a unique contribution for each feature value.

Precisely, IME is based on the idea that the feature values of an instance work together to cause a change in the model's prediction with respect to the model's expected output, and it divides this total change in prediction among the features in a way that is "fair" to their contributions across all possible subsets of features.

2. Cooperative game theory

Definition 2.1. A cooperative (or coalitional) game is a tuple $(\{1, ..., p\}, v)$, where $\{1, ..., p\}$ is a finite set of p players and $v : 2^p \to \mathbb{R}$ is a characteristic function such that $v(\emptyset) = 0$.

Subsets of players $S \subset \{1, ..., p\}$ are *coalitions* and the set of all players $\{1, ..., p\}$ is called the *grand coalition*. Characteristic function v describes the worth of each coalition. We usually assume that the grand coalition forms and the goal is to split its worth (defined by the characteristic function) among the players in a "fair" way. Therefore, the solution is an operator φ which assigns to the game $(\{1, ..., p\}, v)$ a vector of payoffs $\varphi = (\varphi_1, \varphi_2, ..., \varphi_p)$.

For each game with at least one player there are infinite solutions, some of which are more "fair" than others. The following four properties are attempts at axiomatizing the notion of "fairness" of a solution φ:

Axiom 1 (Efficiency). $\sum_{i \in \{1,...,p\}} \varphi_i(v) = v(\{1, ..., p\})$.

AXIOM 2 (Symmetry). *If for two players i and j, $v(S\cup\{i\}) = v(S\cup\{j\})$ holds for every S, where $S \subset \{1,...,p\}$ and $i,j \notin S$, then $\varphi_i(v) = \varphi_j(v)$.*

AXIOM 3 (Dummy). *If $v(S\cup\{i\}) = v(S)$ holds for every S, where $S \subset \{1,...,p\}$ and $i \notin S$, then $\varphi_i(v) = 0$.*

AXIOM 4 (Additivity). *For any pair of games $v, w : \varphi(v+w) = \varphi(v) + \varphi(w)$, where $(v+w)(S) = v(S) + w(S)$ for all S.*

The Shapley value is a "fair" way to distribute the total gains $v(\{1,...,p\})$ to the p players in the sense that it is the only distribution with the four previous desirable properties.

THEOREM 1. For the game $(\{1,...,p\}, v)$ there exists a unique solution φ which satisfies axioms 1 to 4 and it is the Shapley value:

$$Sh_i(v) = \sum_{S\subseteq\{1,...,p\}\setminus\{i\}} \frac{(p-|S|-1)!|S|!}{p!}\Big(v(S\cup\{i\}) - v(S)\Big), \quad i = 1,...,p$$

PROOF. For a detailed proof of this theorem refer to Shapley's paper (1953). □

Moreover, there is an equivalent formula for the Shapley value:

$$\varphi_i(\Delta_{\tilde{x}}) = \frac{1}{p!}\sum_{\mathcal{O}\in S_p}\Big(v\big(Pre^i(\mathcal{O})\cup\{i\}\big) - v\big(Pre^i(\mathcal{O})\big)\Big), \quad i = 1,...,p \tag{2.1}$$

where S_p is the symmetric group of the finite set $\{1,...,p\}$ (i.e., the set of all permutations of the set of players $\{1,...,p\}$) and $Pre^i(\mathcal{O})$ is the set of players which are predecessors of player i in permutation $\mathcal{O} \in S_p$ (that is, the numbers that appear before number i in permutation $\mathcal{O} \in S_p$).

3. Method

Let $\mathcal{X} = X_1 \times X_2 \times ... \times X_p$ be the feature space of p features, represented with the set $\{1,...,p\}$.

Let f be the model being explained. In classification, $f(x)$ is the probability (or a binary indicator) that x belongs to a certain class[1]. In regression, $f(x)$ is the regression function.

Finally, let $\tilde{x} = (\tilde{x}_1,...,\tilde{x}_p) \in \mathcal{X}$ the instance from the feature space whose prediction we want to explain.

Štrumbelj and Kononenko (see References [**7**] and [**8**]) define the *prediction difference* of a subset of feature values in a particular instance as the change in expectation caused by observing those feature values. Formally:

DEFINITION 3.1. Let f be a model. Let $S = \{i_1, i_2, ..., i_s\} \subseteq \{1,...,p\}$ be a subset of features. The prediction difference $\Delta_{\tilde{x}}(S)$ of the subset of features $S \subseteq \{1,...,p\}$ in instance $\tilde{x} = (\tilde{x}_1,...,\tilde{x}_p) \in \mathcal{X}$ is:

$$\Delta_{\tilde{x}}(S) = E\big[f(X_1,...,X_p)|X_{i_1} = \tilde{x}_{i_1},...,X_{i_s} = \tilde{x}_{i_s}\big] - E\big[f(X_1,...,X_p)\big]$$

Note that $\Delta_{\tilde{x}}$ defines a function $\Delta_{\tilde{x}} : 2^p \to \mathbb{R}$ from the set of all possible subsets of features to $\mathbb{R}$ that satisfies $\Delta_{\tilde{x}}(\emptyset) = E\big[f(X_1,...,X_p)\big] - E\big[f(X_1,...,X_p)\big] = 0$. Therefore, $\Delta_{\tilde{x}}$ is a valid characteristic function for the cooperative (or coalitional) game with the p features as players. Therefore, $(\{1,...,p\}, \Delta_{\tilde{x}})$ forms a cooperative game as defined in Def. 2.1. In other words, IME considers features to be players of a game where the worth of the coalitions is the change in the model's prediction

[1]For multiple classes, IME explains each class separately, thus $f(x)$ is the prediction of the relevant class.

defined by $\Delta_{\tilde{x}}$, so the goal of IME is to split the total difference in prediction $\Delta_{\tilde{x}}(\{1,...,p\})$ among the features in a "fair" way.

The suggested contribution of the i-th feature for the explanation of the prediction of $\tilde{x}$ is the Shapley value (see Theorem 1) of the cooperative game $(\{1,...,p\}, \Delta_{\tilde{x}})$:

DEFINITION 3.2. Let {1,...,p} be the set of features and let $\Delta_{\tilde{x}}$ be the prediction difference defined as in Definition 3.1. The contribution of the i-th feature for the explanation of the prediction of $\tilde{x}$ is:

$$\varphi_i(\Delta_{\tilde{x}}) := Sh_i(\Delta_{\tilde{x}}) = \sum_{S \subseteq \{1,...,p\} \setminus \{i\}} \frac{(p - |S| - 1)!|S|!}{p!} \Big(\Delta_{\tilde{x}}(S \cup \{i\}) - \Delta_{\tilde{x}}(S) \Big), \quad i = 1, ..., p$$

Or, equivalently, using the alternative formula for the Shapley value (see Eq. 2.1):

$$\varphi_i(\Delta_{\tilde{x}}) = \frac{1}{p!} \sum_{\mathcal{O} \in S_p} \Big(\Delta_{\tilde{x}}\big(Pre^i(\mathcal{O}) \cup \{i\}\big) - \Delta_{\tilde{x}}\big(Pre^i(\mathcal{O})\big) \Big), \quad i = 1, ..., p \tag{3.1}$$

where S_p is the set of all permutations of the set of features $\{1,...,p\}$ and $Pre^i(\mathcal{O})$ is the set of features which are predecessors of feature i in permutation $\mathcal{O} \in S_p$.

Note that this definition of contributions is not casual: as discussed in section 2, the Shapley value φ of a coalitional game $(\{1,...,p\}, v)$ is a "fair" way to distribute the total gains to the p players. In our case, axioms 1 to 3 become three desirable properties of IME as an explanation method:

1. Efficiency axiom:

$$\sum_{i \in \{1,...,p\}} \varphi_i(\Delta_{\tilde{x}}) = \Delta_{\tilde{x}}(\{1,...,p\}) = E[f|\tilde{x}] - E[f] = f(\tilde{x}) - E[f]$$

That is, the sum of all p contributions in an observation's explanation is equal to the difference between the model's prediction for the instance and the model's expected output given no information about the instance's feature values. This property makes contributions easier to compare across different observations and different models.

2. Symmetry axiom:

If for two features i and j, $\Delta_{\tilde{x}}(S \cup \{i\}) = \Delta_{\tilde{x}}(S \cup \{j\})$ holds for every S, where $S \subset \{1,...,p\}$ and $i, j \notin S$, then $\varphi_i(\Delta_{\tilde{x}}) = \varphi_j(\Delta_{\tilde{x}})$.

In other words, if two features have an identical influence on the prediction, they are assigned identical contributions.

3. Dummy axiom:

If $\Delta_{\tilde{x}}(S \cup \{i\}) = \Delta_{\tilde{x}}(S)$ holds for every S, where $S \subset \{1,...,p\}$ and $i \notin S$, then $\varphi_i(\Delta_{\tilde{x}}) = 0$.

That is to say, a feature is assigned a contribution of 0 if it has no influence on the prediction.

Both the magnitude and the sign of the contributions are important. First, if a feature has a larger contribution than another, it has a larger influence on the model's prediction for the observation of interest. Second, the sign of the contribution indicates whether the feature contributes towards increasing (if positive) or decreasing (if negative) the model's output. Finally, the generated contributions sum up to the difference between the model's output prediction and the model's expected output given no information about the values of the features. In summary, we can discern how much the model's output changes due to the specific feature values for the observation, which features are responsible for this change, and the magnitude of influence of each feature.

In practice, the main challenge of this method is the exponential time complexity because the computation of the contributions requires to use all possible subsets of features. To avoid this issue, Štrumbelj and Kononenko (see Ref. [**7**] and [**8**]) developed an efficient sampling procedure to approximate feature contributions by perturbing the instance's input features.

The sampling procedure developed by Štrumbelj and Kononenko considers $S_p \times \mathcal{X}$ as sampling population, so each order-instance pair $(\mathcal{O}, z)$ defines one sample $T_{\mathcal{O}\in S_p, z\in\mathcal{X}} = f(x') - f(x'')$, where $f(x')$ and $f(x'')$ are the model's predictions for two observations x' and x'' constructed by taking instance z and then changing the value of each feature which appears before the i-th feature in order $\mathcal{O}$ (for x' this includes the i-th feature) to that feature's value in $\tilde{x}$. That is:

$$x'_k = \begin{cases} \tilde{x}_k \text{ , if } k \in Pre^i(\mathcal{O}) \cup \{i\} \\ z_k \text{ , if } k \notin Pre^i(\mathcal{O}) \cup \{i\} \end{cases} \quad \text{and} \quad x''_k = \begin{cases} \tilde{x}_k \text{ , if } k \in Pre^i(\mathcal{O}) \\ z_k \text{ , if } k \notin Pre^i(\mathcal{O}) \end{cases}$$

According to Štrumbelj and Kononenko, $E[T_{\mathcal{O}\in S_p, z\in\mathcal{X}}] = \varphi_i$ if we draw samples completely at random. Thus, if N such samples are drawn (with replacement) and observe the random variable $\hat{\varphi}_i = \frac{1}{N}\sum_{j=1}^{N} T_j$, where T_j is the j-th sample. According to the central limit theorem, $\hat{\varphi}_i$ is approximately normally distributed with mean φ_i and variance $\frac{\sigma_i^2}{N}$, where σ_i^2 is the population variance for the i-th feature. Thus, $\hat{\varphi}_i$ is an unbiased and consistent estimator of φ_i. The computation is summarized in Algorithm 1.

Algorithm 1 Approximating the contribution of the i-th feature's value φ_i for instance $\tilde{x} \in \mathcal{X}$ and model f

Require: model f, instance $\tilde{x}$
Require: number of samples N
1: $\varphi_i \leftarrow 0$
2: **for** $j = 1$ to N **do**
3: choose a random permutation of features $\mathcal{O} \in S_p$
4: choose a random instance $z \in \mathcal{X}$
5: $x' \leftarrow$ **if** $k \in Pre^i(\mathcal{O}) \cup \{i\}$ **then** $x'_k = \tilde{x}_k$ **else** $x'_k = z_k$
6: $x'' \leftarrow$ **if** $k \in Pre^i(\mathcal{O})$ **then** $x''_k = \tilde{x}_k$ **else** $x''_k = z_k$
7: $\varphi_i \leftarrow \varphi_i + f(x') - f(x'')$
8: $\varphi_i \leftarrow \frac{\varphi_i}{N}$

4. Example

Let X_1, X_2 be two binary features, $C = \{0, 1\}$ a binary class, and $\mathcal{X} = \{0, 1\} \times \{0, 1\}$ our feature space. We assume that all feature value combinations are equiprobable, as seen in Table 1.

Assume that the class is defined by the disjunction $C = X_1 \vee X_2$, that is, $C = 1$ if $X_1 = 1$ or $X_2 = 1$, and 0 otherwise. Let $f : \mathcal{X} \to [0, 1]$ be an (ideal) classifier.

We will compute the explanation produced by IME to explain the prediction for the observation $\tilde{x} = (1, 0)$ from the perspective of class value 1. In this case, we can compute analytically the contribution of each feature. Specifically, we will use the alternative formula for the Shapley value (see 3.1).

First, we need to compute the symmetric group S_2 of the finite set $\{1, 2\}$, which has $2! = 2$ elements:

$$S_3 = \{\{1, 2\}, \{2, 1\}\}$$

X_1	X_2	$C(= f(X))$
0	0	0
1	0	1
0	1	1
1	1	1

TABLE 1. A simple data set used for illustrative purposes. All combinations of features are equally probable.

Second, we have to compute the prediction differences (see Def. 3.1) for all subsets of features, so we first compute the expected prediction if no feature values are known:

$$E\big[f(X_1, X_2)\big] = \sum_{(x_1,x_2)\in\mathcal{X}} f(x_1, x_2) \cdot P(X_1 = x_1, X_2 = x_2) = \frac{1}{4}(0 + 1 + 1 + 1) = \frac{3}{4}$$

Now we can compute the prediction differences:

$\Delta_{\tilde{x}}(\emptyset) = 0$

$\Delta_{\tilde{x}}(\{1\}) = E\big[f(X_1, X_2)|X_1 = 1\big] - E\big[f(X_1, X_2)\big] = \frac{(1+1)}{2} - \frac{3}{4} = 1 - \frac{3}{4} = \frac{1}{4}$

$\Delta_{\tilde{x}}(\{2\}) = E\big[f(X_1, X_2)|X_2 = 0\big] - E\big[f(X_1, X_2)\big] = \frac{(1+0)}{2} - \frac{3}{4} = \frac{1}{2} - \frac{3}{4} = -\frac{1}{4}$

$\Delta_{\tilde{x}}(\{1, 2\}) = E\big[f(X_1, X_2)|X_1 = 1, X_2 = 0\big] - E\big[f(X_1, X_2)\big] = \frac{1}{1} - \frac{3}{4} = 1 - \frac{3}{4} = \frac{1}{4}$

Finally, we can compute the contributions of the features:

$\varphi_1 = \frac{1}{2!}\Big[\Big(\Delta_{\tilde{x}}\big(Pre^1(\{1,2\})\cup\{1\}\big) - \Delta_{\tilde{x}}\big(Pre^1(\{1,2\})\big)\Big) + \Big(\Delta_{\tilde{x}}\big(Pre^1(\{2,1\})\cup\{1\}\big) - \Delta_{\tilde{x}}\big(Pre^1(\{2,1\})\big)\Big)\Big] =$
$\frac{1}{2}\Big[\Big(\Delta_{\tilde{x}}\big(\emptyset\cup\{1\}\big) - \Delta_{\tilde{x}}(\emptyset)\Big) + \Big(\Delta_{\tilde{x}}\big(\{2\}\cup\{1\}\big) - \Delta_{\tilde{x}}\big(\{2\}\big)\Big)\Big] = \frac{1}{2}\Big[\Big(\Delta_{\tilde{x}}\big(\{1\}\big) - \Delta_{\tilde{x}}(\emptyset)\Big) + \Big(\Delta_{\tilde{x}}\big(\{1,2\}\big) -$
$\Delta_{\tilde{x}}\big(\{2\}\big)\Big)\Big] = \frac{1}{2}\Big[\Big(\frac{1}{4} - 0\Big) + \Big(\frac{1}{4} - (-\frac{1}{4})\Big)\Big] = \frac{3}{8}$

$\varphi_2 = \frac{1}{2!}\Big[\Big(\Delta_{\tilde{x}}\big(Pre^2(\{1,2\})\cup\{2\}\big) - \Delta_{\tilde{x}}\big(Pre^2(\{1,2\})\big)\Big) + \Big(\Delta_{\tilde{x}}\big(Pre^2(\{2,1\})\cup\{2\}\big) - \Delta_{\tilde{x}}\big(Pre^2(\{2,1\})\big)\Big)\Big] =$
$\frac{1}{2}\Big[\Big(\Delta_{\tilde{x}}\big(\{1\}\cup\{2\}\big) - \Delta_{\tilde{x}}\big(\{1\}\big)\Big) + \Big(\Delta_{\tilde{x}}\big(\emptyset\cup\{2\}\big) - \Delta_{\tilde{x}}\big(\emptyset\big)\Big)\Big] =$
$= \frac{1}{2}\Big[\Big(\Delta_{\tilde{x}}\big(\{1,2\}\big) - \Delta_{\tilde{x}}\big(\{1\}\big)\Big) + \Big(\Delta_{\tilde{x}}\big(\{2\}\big) - \Delta_{\tilde{x}}\big(\{\emptyset\}\big)\Big)\Big] = \frac{1}{2}\Big[\Big(\frac{1}{4} - \frac{1}{4}\Big) + \Big(-\frac{1}{4} - 0\Big)\Big] = -\frac{1}{8}$

The contribution of the first feature is positive while the contribution of the second feature is negative, which is reasonable. Since the class is 1 when at least one of the features is 1, the first feature of our instance made the model predict 1, so this value contributed to the model's decision of predicting 1. On the contrary, the second feature made less probable that 1 was the predicted class, so it had a negative influence on the model's decision. Additionally, the absolute value of the first contribution is larger than the second one, which means that the first feature was decisive for the prediction. Finally, note that the two contributions sum up to the initial difference between the prediction for this instance and the expected prediction when the feature values are unknown: $\varphi_1 + \varphi_2 = \frac{3}{8} - \frac{1}{8} = \frac{1}{4} = 1 - \frac{3}{4} = f(1, 0) - E[f(X_1, X_2)]$, which was assured by axiom 1.

In summary, the explanation produced by IME for the prediction of the instance $(1, 0)$ is that the model's decision is influenced by both features, being the first feature more decisive in favor of (correctly) predicting class 1 than the second feature, whose value was against this decision.

Part 2

Experiment

In this part we will apply the methods described in Part 1 to explain predictions of machine learning models. To do so, we will train models[2] on a simple data set, explain some of their predictions and compare the explanations with our previous knowledge of the problem. Specifically, there are several questions that we will try to answer with the experiment:

(1) Do the explanation methods produce useful explanations for predictions made by machine learning models (including black box models)?
(2) Do the explanation methods really produce different explanations for different predictions?
(3) Do the three explanation methods produce similar explanations for the same prediction made by a model?
(4) Are the explanation methods really useful to compare models?

First, we will start Chapter 5 by describing the data set used for the experiment (section 1) and by detailing the followed methodology (section 2). Second, four predictions made by one specific trained model will be explained using the three studied explanation methods (section 3) and the explanations produced for one of these four predictions will be used to compare the explanation methods (section 4). Finally, we will compare the explanations produced by the three methods to explain the prediction of four different machine learning models for the same instance (section 5).

The experiment has been run in R. Regarding the explanation methods, while LIME and IME are currently implemented in R (see References [**11**] and [**12**], respectively), explanation vectors are not. For this thesis, we have developed our own functions to explain and visualize predictions of any model, taking advantage of the LIME implementation but not the IME one because it only supports models implemented in package CORElearn by now. Finally, the caret package (see Reference [**13**]) is used to train and tune hyperparameters of all models and the ggplot2 package (see Reference [**14**]) is used for all data visualizations.

[2]Note that the focus of the experiments is the application of LIME, explanation vectors and IME instead of the training process of models, so the models' hyperparameters will not be thoroughly optimized.

Chapter 5
Cylinder in a cube

1. Description

The problem is a binary classification task in three dimensions. All variables (x_1, x_2, x_3) are uniformly distributed on the interval $[-1, 1]$, so the data points form a cube. Class 1 is a cylinder in the middle of the cube whose section is a circle defined by the first two variables. The remainder forms class 2. The radius of the cylinder is chosen such that both classes have equal prior probability 0.5. An example of the resulting dataset is shown in Figure 2.

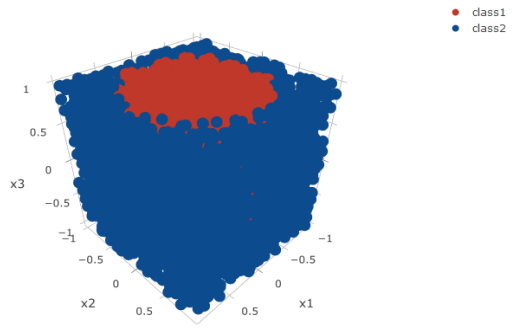


FIG. 1. Example of the dataset used for the experiment. The classes form a cylinder (red points) in a cube (blue points).

.

In summary, it is a binary classification task with two features (x_1 and x_2) which determine the class and a meaningless one (x_3) that does not have any influence in the actual class label of data points.

2. Methodology

We created two independent sets: a training set of 1,000 observations (see Figure 2) and a test set of 500 observations. Models are trained on the training set of 1,000 examples. 5-fold cross validation is used for parameter tuning and model selection, except for the random forest model, for which the out-of-bag error is used instead. The test set is used to both compute the accuracy of the trained models and choose observations that are representative of the problem to be explained with the three explanation methods described in Part 1.

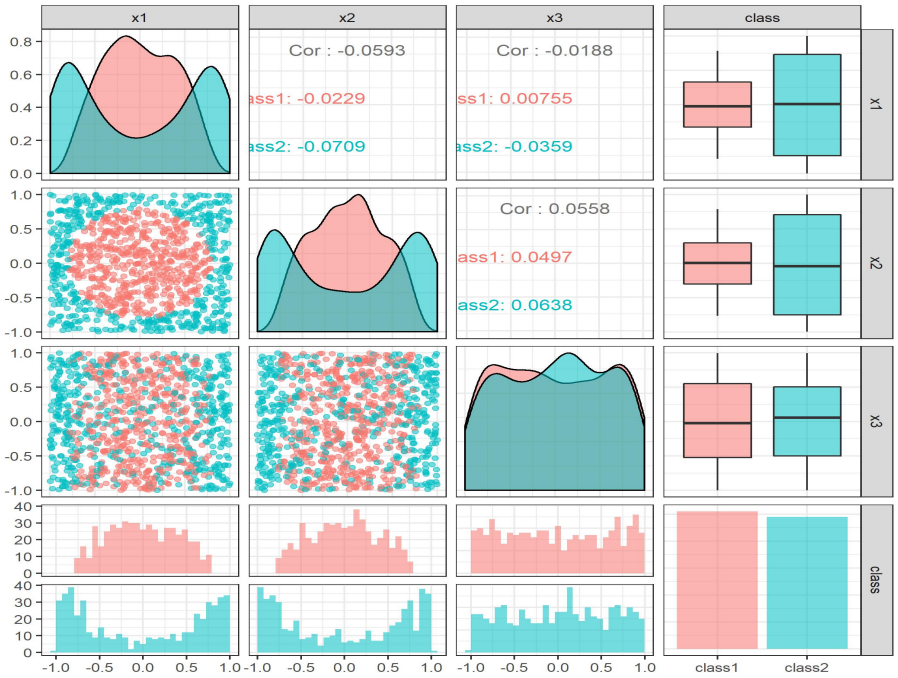


FIG. 2. Pairs plot of the training set. Each element of the matrix of plots displays the relationship between two variables in the dataset.

.

For LIME explanations, we produced linear explanations using $K = 3$, that is, using all features. In this case, since explanations composed of three parts are understandable for humans, there is no need to reduce the number of features used in the explanations[1]. Furthermore, we used as *interpretable representation* the resulting from converting the three continuous variables into three binary variables that specify whether an observation belongs to the same quantile (defined according to the training data) of the instance being explained or not. Since we want to explain each instance separately, the kernel width σ and the number of bins used to convert the original variables into binary are optimized separately for each observation based on the coefficient of determination R^2 of the linear model obtained as explanation, which is a measure of the quality of the explanation. Although this is not a suggestion by the Ribeiro et al. , we think that this way we use the "best" hyperparameters to explain each obervation instead of a fixed set of hyperparameters for all instances which could not be useful for specific instances and lead to wrong explanations. Finally, we used $N = 10,000$ samples.

For explanation vectors, there is only a hyperparameter σ if we want to explain a hard classifier and we have to estimate the explanation vectors using a kernel density estimator. In that case, we optimized the value of the kernel width σ using the test data as described in Chapter 3.

For IME, we used the sampling algorithm described in Chapter 4 (Section 3) with $N = 10,000$.

3. Explaining predictions

In this section, we will explain four predictions made by an artifficial neural network from the perspective of class 1 and using the three explanation methods: LIME (see Chapter 2), explanation

[1]However, in high dimensional problems we should choose a small number of features K in order to have an understandable explanation. The number of features could be optimized using the coefficient of correlation R^2 of the explanation model.

vectors (see Chapter 3) and IME (see Chapter 4). In order to facilitate the explanation process, we will visualize the explanations produced by any of the three methods with an horizontal bar chart that represents the contribution to the prediction of each considered variable, a common approach found in the literature (see References [**1**] or [**7**]). In addition, bars are green for features whose values influence the model in favor of the class being explained, and red for feature whose values influence the model against the class being explained. An example of this visualization is Figure 4.

The final neural network after tuning the hyperparameters has a single hidden layer with 5 neurons and decay parameter $\lambda = 10^{-5}$. The accuracy of this model is 98.8% on the test data.

The instances that we will explain are four representative points of the problem from the test set:

(1) Observation 1: a point close to the border of the cylinder with $|x_1| \approx |x_2|$, $\tilde{x}^{(1)} = (-0.53, 0.56, 0.3)$
(2) Observation 2: a point close to the border of the cylinder with $x_1 \approx 0$, $\tilde{x}^{(2)} = (-0.06, 0.79, -0.96)$
(3) Observation 3: a point close to the axis of the cylinder, $\tilde{x}^{(3)} = (-0.17, 0.06, 0.99)$
(4) Observation 4: a point close to the border of the cylinder with $x_2 \approx 0$, $\tilde{x}^{(4)} = (0.79, 0.11, -1.19)$

The first two variables (the meaningful ones) of these observations can be seen in Figure 3 along with the contour map of the probability function f learned by the neural network when $x_3 = 0$. All these observations are of actual class 1. The neural network is able to correctly classify observations 1 to 3, but it incorrectly predicts class 2 for observation 4. Specifically, the predicted probabilities (from the perspective of class 1) are: 0.9999820, 0.6556746, 1.0 and 0.1322022.

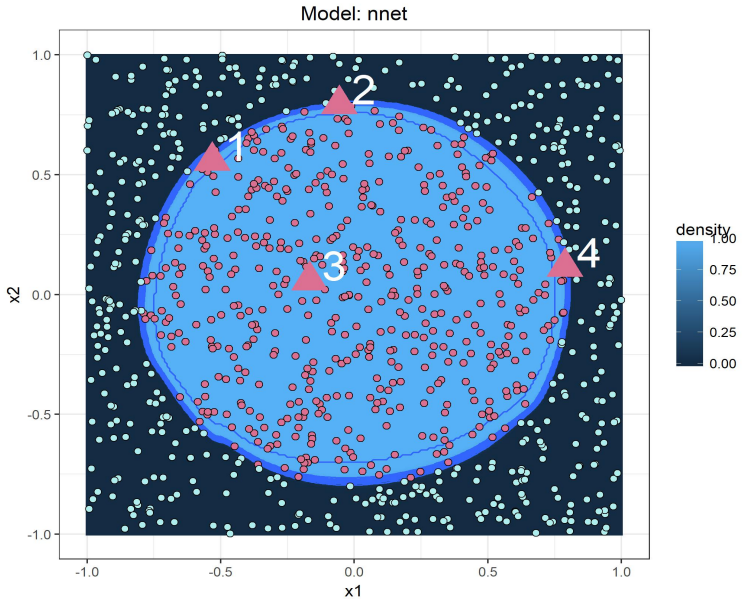


FIG. 3. Scatterplot of x_2 against x_1 of the training data (circular points) overlaid on the contour map of the probability function f learned by a neural network when $x_3 = 0$. Points inside the circle belong to class 1 (in violet) while points outside the circle belong to class 2 (in turquoise). Also, four test points (triangular points) are displayed.

Theoretically, if the model were perfect, we would expect that the contribution of x_3 to any prediction was 0, while the contributions of x_1 and x_2 should be different depending on their values. First, it is reasonable to think that the prediction for observation 1 should take into account both x_1 and x_2 in order to make a correct prediction because the combination of the two makes it more likely to belong to class 1 than other possible combinations of the variables. Second, the value of x_2 of

observation 2 is extreme for a point of actual class 1, but the value of x_1 makes it possible. Therefore, the correct explanation for this prediction is that both features are important but in different ways: while the value of x_1 supports that the class is 1, the value of x_2 contradicts this possibility. Third, observation 3 is clearly from class 1 (because it is inside the cylinder) and both x_1 and x_2 have typical values of class 1, so both contribute to be class 1. Finally, observation 4 is comparable to observation 2 but with the opposite roles of variables 1 and 2, so the explanation for its prediction is that, despite the fact that x_1 gives arguments against class 1, the actual class is 1 thanks to x_2.

In the remaining part of this section we will discuss the explanations produced by the studied explanation methods and check if they are similar to the theoretical ones described in the last paragraph. The latter would be a first step to determine if the trained neural network has correctly learned when a data point belongs to class 1. However, in order to know for certain that we can *trust* this model, we would need to check explanations for much more predictions made by the model. At this point, it should be remembered that the explanations that we will obtain are specific for this neural network model and they could be completely different if we had used a different model because the learned probability function f would be different.

3.1. LIME. The explanations produced by LIME are shown in Figure 4. First, we see that the method produces different explanations for different predictions, that is, it is able to determine that the model had different reasons behind each prediction. Second, we see that the explanations are in a binary *interpretable representation*, so this method has the advantage that it highlights the range of values of each feature that really has influence on the model's prediction. For instance, stating that "the fact that x_1 has a value smaller than -0.4927 gives arguments against predicting class 1" is more informative than just stating that "the value of x_1 gives arguments against predicting class 1". Finally, it is worth noting that LIME would be able to produce simple (and short) explanations even in high dimensional problems because we can decide the number of variables K to use for the explanations. Although the simple 3-dimensional data set that we are using in this chapter allows us to keep all three variables (i.e., $K = 3$), this could make a big difference if our data set had hundreds or thousands of variables. However, the flexibility of LIME has the drawback that there are several hyperparameters that need to be set: the number of features K to use for each explanation, the kernel width σ that determines the weight function to define the locality around the instance being explained and, in our case of tabular data, the number of groups used in the interpretable representation defined by converting continuous features into binary variables.

The best explanation found for observation 1 has $\sigma = 1$ and the continuous features are split into 4 groups. The coefficient of correlation of the interpretable linear model obtained, which can be interpreted as a measure of the explanation's quality, is $R^2 = 0.19$. The interpretable linear model obtained is:

$$g_{\tilde{x}^{(1)}}(x) = \hat{f}_{\tilde{x}^{(1)}}(x) = 0.663 - 0.314\ \mathbb{1}_{x_1 \leq -0.4927}(x) - 0.31\ \mathbb{1}_{x_2 > 0.49097}(x) - 0.011\ \mathbb{1}_{0.0108 < x_3 \leq 0.5371}(x)$$

That is, the contributions of the features are: -0.314 for x_1, -0.31 for x_2 and -0.011 for x_3. According to LIME, since the absolute value of the latter is very small compared to the other two, we can conclude that the model's decision of predicting class 1 is influenced by both x_1 and x_2, but not by x_3. Specifically, both meaningful features have the same influence on the prediction because they have very similar absolute values and the sign of their contributions is negative, which means that their values reduce the probability of predicting class 1. This is reasonable because observation 1 is a data point with values of x_1 and x_2 quite large for an observation that belongs to class 1. It may seem counterintuitive that both meaningful variables gives arguments against predicting class 1 but the final prediction of the neural network is still class 1. This happens because the model intercept is larger than 0.5, which means that instances close to the observation (recall that LIME weights by the distance from the observation being explained) but that do not belong to any of the quartiles of the observation (recall that the explanation model is fitted using a binary interpretable representation) are more likely to belong to class 1. Therefore, the explanation for this prediction would be that

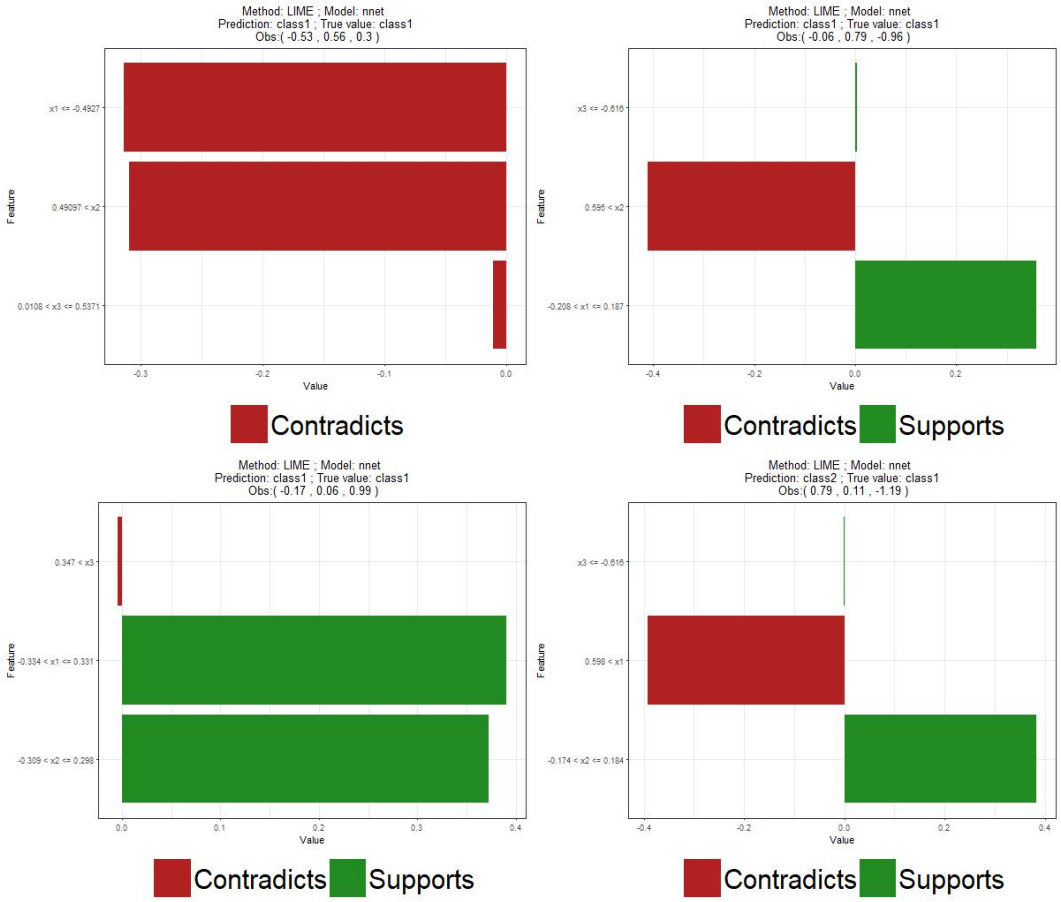


FIG. 4. LIME explanations for the predictions produced by a neural network for four observations: $\tilde{x}^{(1)} = (-0.53, 0.56, 0.3)$ (upper left), $\tilde{x}^{(2)} = (-0.06, 0.79, -0.96)$ (upper right), $\tilde{x}^{(3)} = (-0.17, 0.06, 0.99)$ (bottom left) and $\tilde{x}^{(4)} = (0.79, 0.11, -1.19)$ (bottom right).

the neural network (correctly) predicts class 1 for this observation because the given evidence against predicting this class by the fact that $\tilde{x}_1^{(1)} \leq -0.4927$ and $\tilde{x}_2^{(1)} > 0.49097$ is not strong enough to change the predicted class of similar instances. This explanation coincides with the knowledge that we have of this data set and that we discussed in the previous section, which is an indication that the model has correctly learned the underlying function that determines the relationship between the three variables and the class.

The best explanation found for observation 2 has $\sigma = 1$ and the features are split into 5 groups. The coefficient of correlation of the interpretable linear model obtained is $R^2 = 0.25$. The interpretable linear model obtained is:

$$g_{\tilde{x}^{(2)}}(x) = \hat{f}_{\tilde{x}^{(2)}}(x) = 0.52 + 0.358\ \mathbb{1}_{-0.208 < x_1 \leq 0.187}(x) - 0.41\ \mathbb{1}_{x_2 > 0.595}(x) + 0.004\ \mathbb{1}_{x_3 \leq -0.616}(x)$$

Again, the model's decision of predicting class 1 is influenced by both x_1 and x_2, but in different ways and being the role of x_2 a little more important than the one of x_1. On the one hand, the value of x_1 supports that the class is 1 because its contribution has positive sign, which means that having the value on the interval $(-0.208, 0.187)$ increases the probability of predicting class 1. On the other hand, the value of x_2 gives arguments against predicting the class is 1 because its value is

on the interval $(-0.208, 0.187)$ and this reduces the probability of predicting class 1. As we discussed before, this is reasonable because the value of x_2 of observation 2 is extreme for a point of actual class 1. Thus, the explanation for this prediction would be that the model (correctly) predicts class 1 for observation 2 because $-0.208 < \tilde{x}_1^{(2)} \leq 0.187$, despite the fact that $\tilde{x}_2^{(2)} > 0.59$ is an abnormally large value for a data point of class 1. This explanation also coincides with the knowledge that we have of this data set, which is another indication that the model has correctly learned the true probability function.

The best explanation found for observation 3 has $\sigma = 1$ and the features are split into 3 groups. The coefficient of correlation of the interpretable linear model obtained is $R^2 = 0.31$. The interpretable linear model obtained is:

$$g_{\tilde{x}^{(3)}}(x) = \hat{f}_{\tilde{x}^{(3)}}(x) = 0.263 + 0.39 \ \mathbb{1}_{-0.334<x_1\leq 0.331}(x) + 0.372 \ \mathbb{1}_{-0.309<x_2\leq 0.298}(x) - 0.004 \ \mathbb{1}_{x_3>0.347}(x)$$

In other words, both x_1 and x_2 has contributed to correctly predict class 1. The sign of both feature contributions are positive, that is, the values of these two features increase the probability of predicting class 1. Therefore, both values support that the class is 1, which is a correct reasoning because this observation is clearly inside the cylinder that defines this class. In summary, LIME's explanation for this prediction would be as simple as that the model predicts class 1 because $-0.334 < \tilde{x}_1^{(3)} \leq 0.331$ and $-0.309 < \tilde{x}_2^{(3)} \leq 0.298$. Again, this reasoning coincides with the knowledge that we have of this data set.

The best explanation found for observation 4 has $\sigma = 1$ and the features are split into 5 groups. The coefficient of correlation of the interpretable linear model obtained is $R^2 = 0.26$. The interpretable linear model obtained is:

$$g_{\tilde{x}^{(4)}}(x) = \hat{f}_{\tilde{x}^{(4)}}(x) = 0.506 - 0.391 \ \mathbb{1}_{x_1>0.598}(x) + 0.383 \ \mathbb{1}_{-0.174<x_2\leq 0.184}(x) + 0.001 \ \mathbb{1}_{x_3\leq -0.616}(x)$$

In this case, the neural network incorrectly predicts class 2 for an instance of actual class 1 due to the influence of both x_1 and x_2. The contribution of the first feature has a negative sign, so this feature gives arguments against predicting class 1 because its actual value decreases the probability of predicting class 1, which is reasonable because the value of x_1 is very large for an instance of this class. On the contrary, the contribution of the second feature has a positive sign, so its value increases the probability of predicting class 1. Furthermore, we can see that the negative contribution has a larger absolute value than the positive one, which justifies that the model finally predicts class 2. In a nutshell, even though $-0.174 < \tilde{x}_2^{(4)} \leq 0.184$ is in favour of predicting class 1, the fact that $\tilde{x}_1^{(4)} > 0.598$ is so unusual for class 1 that the model incorrectly predicts class 2. However, note that this is an understandable mistake because observation 4 is really close to the border of the cylinder (see Figure 3), so at least for now we cannot conclude that this model is untrustworthy. Therefore, this is an example where an explanation method can be used to justify the mistake of a model, which wrongly predicts an observation but for justified reasons.

In conclusion, we have been able to properly explain individual predictions made by a neural network using LIME. Moreover, the previous results indicate that this model has correctly learned the underlying function that determines the relationship between the three variables and the class because all four explanations agree on the theoretical reasons discussed in the previous section that determines the class of the analyzed observations.

3.2. Explanation vectors. The components of the four explanation vectors obtained are shown in Figure 5. Again, the method produces different explanations for different predictions. However, there is not a meaningful explanation for observation 3 because the gradient at that data point (i.e., the

explanation vector) is the zero vector. As we discussed in Chapter 3, this is an issue of this method that appears when the probability function f learned by the model is flat in some neighborhood of the instance being explained. We can see in Figure 3 that this is the case of observation 3 because it is inside the cylinder and the model is completely certain on the class for this data point, that is, the probability function f is probably completely flat (equal to 1) around this instance. Moreover, note that the range of the contributions is very different across observations because the influence on the model's decision is different depending on the position in the feature space.

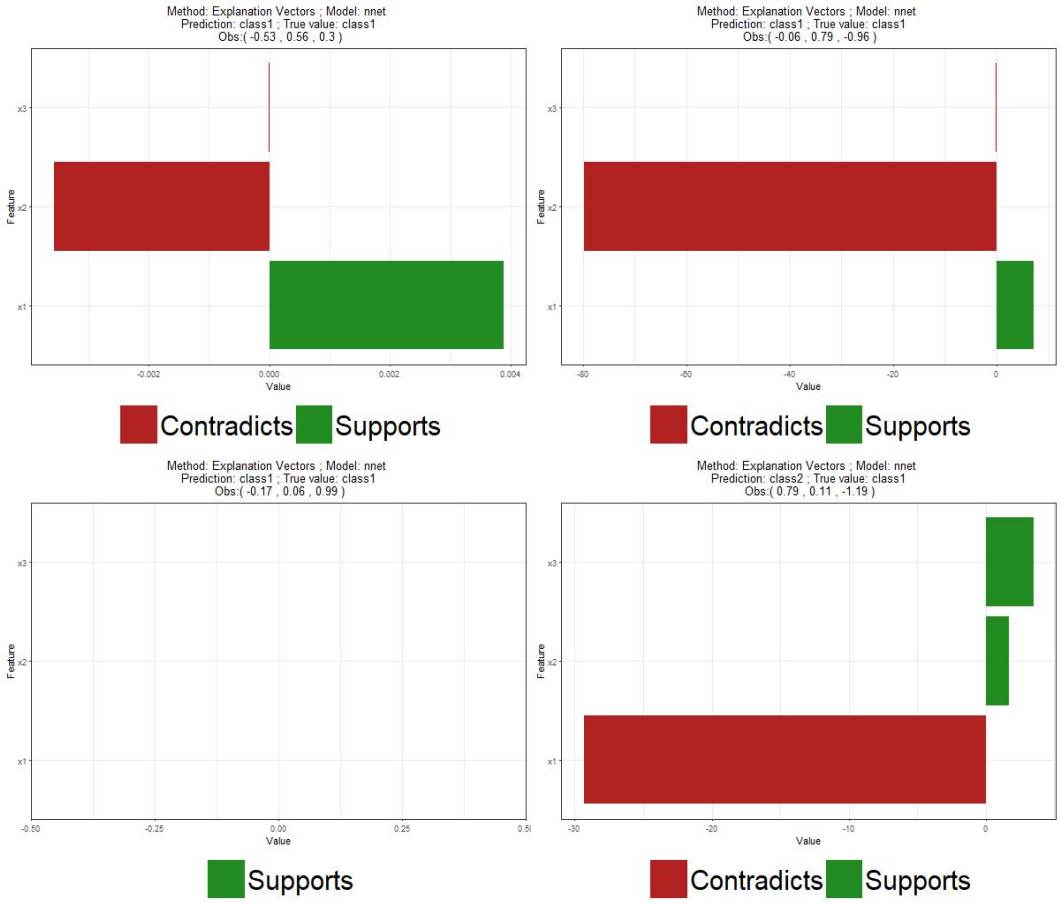


FIG. 5. Explanation vectors for the predictions produced by a neural network for four observations: $\tilde{x}^{(1)} = (-0.53, 0.56, 0.3)$ (upper left), $\tilde{x}^{(2)} = (-0.06, 0.79, -0.96)$ (upper right), $\tilde{x}^{(3)} = (-0.17, 0.06, 0.99)$ (bottom left) and $\tilde{x}^{(4)} = (0.79, 0.11, -1.19)$ (bottom right).

The model's decision of (correctly) predicting class 1 for observation 1 is influenced by both x_1 and x_2, but not by x_3. Since the explanation vector in binary classification tasks is the gradient of the predicted probability for class 1, the explanation for observation 1 means that the model would be more certain of predicting class 1 if the first feature had a larger value and the second value had a smaller one. This is reasonable because, if we move this data point move towards larger values of x_1 and smaller values of x_2, we would be going to the axis of the cylinder in order to be more certain of predicting class 1, so it is an indication that the model has correctly learned the relationship between the variables and the response. Furthermore, the absolute value of the contribution of x_2 is larger than the one of x_1, so the second variable has more influence on the model's prediction.

For observation 2, the model's prediction is mainly influenced by the second feature. Specifically, the sign of its contribution is negative, which means that a smaller feature value would increase the probability of predicting class 1. This is reasonable for the same reason as before: if we decreased the value of x_2, we would be going to the axis of the cylinder, where the model seems to be more certain of predicting class 1. However, in contrast to what we discussed in the previous section, this explanation does not consider that x_1 had almost any influence on the model's output, which seems to be a little simplistic.

Finally, the neural network incorrectly predicts class 2 for observation 4 mainly influenced by the value of x_1. The contribution of this feature is negative, so its value should be smaller in order to increase the predicted probability of class 1 and, eventually, make the model change its current prediction to the correct one. Once again, this makes sense because the explanation vector points towards the axis of the cylinder, where the model seems to be more certain of predicting class 1, but it is a little simplistic because it does not give any credit to x_2 in contrast to what we expected from our knowledge of the data set. Actually, the contribution of the noisy variable x_3 is considered to be more influential than x_2, which could be an indication of an overfitted model.

In conclusion, we have been able to explain individual predictions made by a neural network using explanation vectors. Moreover, the previous results indicate that this model has correctly learned the underlying function that determines the relationship between the three variables and the class because all four explanation vectors, which are local gradients of the learned probability function, point towards the axis of the cylinder as the location where the predicted probability for class 1 is larger, which is theoretically true. However, compared to the theoretical explanations that we discussed in the previous section, the explanations produced by this method seems to be a little simplistic sometimes.

3.3. IME. The explanations produced by IME are shown in Figure 6. Like the other two methods, IME produces different explanations for different predictions.

While the average prediction of the model on the training set is 0.51, the predicted probability produced by the model for observation 1 is 0.9999 (so it correctly predicts the class of the instance), which can be seen as an increase in probability of 0.4899 produced by the cooperation between feature values. The explanation gives credit to both x_1 and x_2, while x_3 is irrelevant for the prediction. Both feature values (-0.53 and 0.56, respectively) contributed in favor of predicting class 1 because the sign of their contributions is positive, which means that knowing one of these two feature values tends to increase the predicted probability with respect to the average prediction. Also, we can extract from the explanation that x_1 has influenced a little more than x_2. Finally, note that the sum of the three contributions is almost equal to the increase in probability with respect to the average prediction: $0.2565 + 0.2231 + 0.001 = 0.4806318 \approx 0.4899$, which was assured by the efficiency axiom described in Section 3. The small difference is due to the approximation algorithm of IME explanations that we have used. This explanation coincides with the knowledge that we have of this data set and that we discussed in the previous section, which is an indication that the model has correctly learned the underlying function that determines the relationship between the three variables and the class.

The predicted probability produced by the model for observation 2 is 0.66, which means that the feature values work together to cause an increase in probability of 0.15. In particular, the model's decision of predicting class 1 is influenced by both x_1 and x_2, but in different ways. On the one hand, the value of x_1 supports that the class is class 1 because its contribution has positive sign, which means that having its value contributes positively to the increase with respect to the average prediction for class 1. On the other hand, the value of x_2 gives arguments against predicting class 1 because it is more characteristic of points from the other class. As we discussed before, this is reasonable because the value of x_2 of observation 2 is extreme for a point of actual class 1. Thus, the explanation for this prediction would be that the model (correctly) predicts class 1 for observation 2 because of the value of x_1, despite the fact that x_2 is an abnormally large value for a data point of class 1. Also, we can extract from the explanation that x_1 is more influential than the one of x_1. This explanation

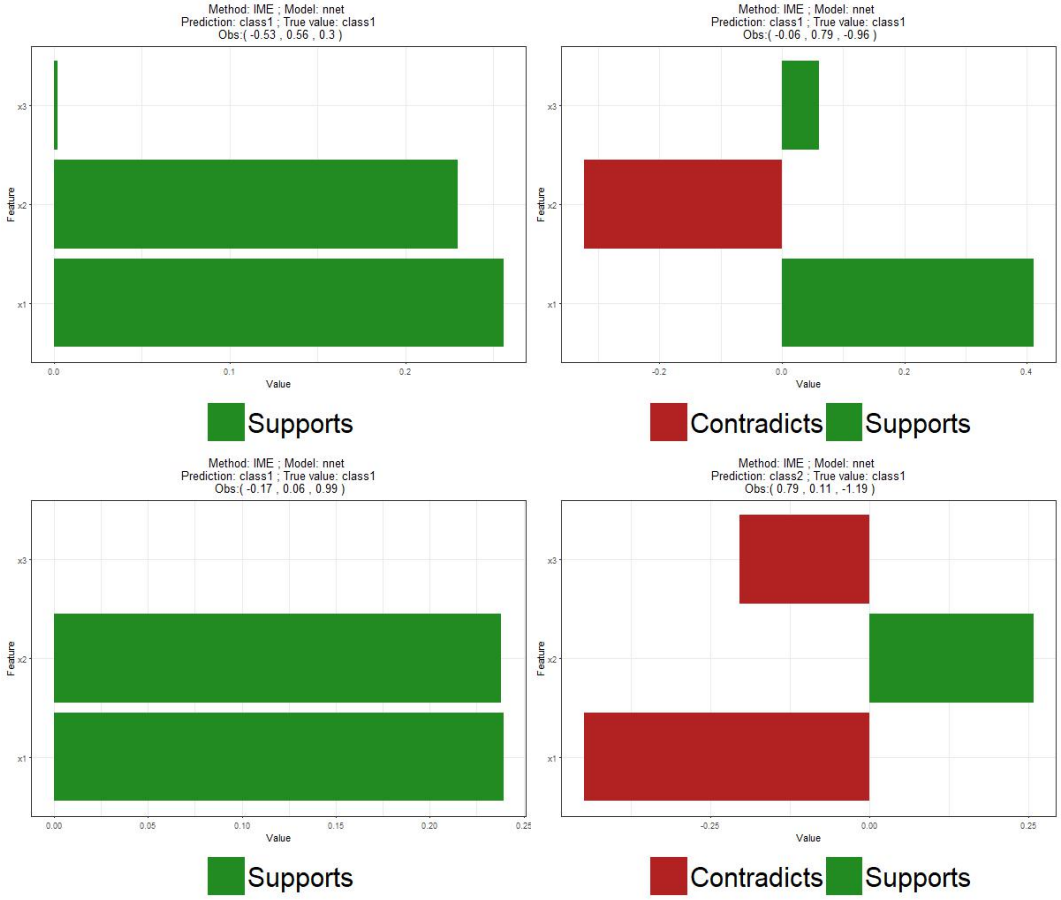


FIG. 6. IME explanations for the predictions produced by a neural network for four observations: $\tilde{x}^{(1)} = (-0.53, 0.56, 0.3)$ (upper left), $\tilde{x}^{(2)} = (-0.06, 0.79, -0.96)$ (upper right), $\tilde{x}^{(3)} = (-0.17, 0.06, 0.99)$ (bottom left) and $\tilde{x}^{(4)} = (0.79, 0.11, -1.19)$ (bottom right).

also coincides with the knowledge that we have of this data set, which is another indication that the model has correctly learned the true probability function.

The predicted probability produced by the model for observation 3 is 1.0, which means that the increase in probability of 0.49 is produced by the cooperation between feature values. Specifically, the model's prediction is influenced by the values of both x_1 and x_2, which turn out to equally contribute because their contributions have almost the same absolute value. Moreover, both contributions are positive, so these two specific feature values tend to increase the probability of predicting class 1, which is reasonable because this observation is clearly inside the cylinder that defines this class. Again, this reasoning coincides with the knowledge that we have of this data set.

Finally, the predicted probability of class 1 for observation 4 is 0.1322, so the model incorrectly predicts the class. In this case, the model's prediction is influenced by the three variables in order to predict class 2. On the one hand, the contribution of x_1 is positive because its value is abnormally large for an observation of class 1, which pushes the model into predict the wrong class. On the other hand, the contribution of x_2 is positive, so it supports that the class is class 1, which is reasonable because the most likely class given its feature value (-0.06) is "class1". Finally, although it has less relevance than the other two features due to its smaller absolute value, variable x_3 contradicts predicting class

1. Consequently, it seems that the model has slightly overfitted the training data. In summary, the explanation for this prediction would be that, even though the value of x_2 gives arguments in favor of predicting class 1, the value of x_2 is so unusually large that makes the model predict the wrong class.

In conclusion, we have been able to properly explain individual predictions made by a neural network using IME. Furthermore, the previous results indicate that this model has correctly learned the underlying function that determines the relationship between the three variables and the class because all four explanations agree on the theoretical reasons discussed in the previous section that determines the class of the analyzed observations.

4. Comparing methods

In this section, we will compare the explanations produced by LIME, explanation vectors and IME for the predictions of the previous section. Each column of plots of Figure 7 shows, from left to right, the plots from Figures 4, 5 and 6, respectively.

At first sight, it seems that the three explanation methods agree on the fact that the third variable is almost irrelevant for the decision process of the neural network, but in essence they seem to produce different explanations for the same prediction made by this model. LIME and IME sometimes produce similar explanations (e.g., observations 2 and 4), and other times they assign opposite roles to the features (e.g., observations 1 and 3). Furthermore, the explanation vectors seems to be always different from the explanations produced by LIME and IME.

These differences between methods are completely understandable given the fact that they focus on different characteristics to determine the influence of features on a prediction. First, LIME's contributions are the coefficients of a weighted linear model that locally approximates the original neural network, so a positive sign of the coefficient indicates that the predicted probability of class 1 increases due to the value of the corresponding explanatory variable while a negative sign indicates that it decreases. Second, explanation vectors' contributions have the same interpretation as LIME's, but note that the explanations of these two methods are completely different because LIME uses a binary interpretable representation instead of the original continuous features. Third, IME's contributions are a measure of the effect that a feature value has across all possible subsets of features in order to change the average prediction (in practice, this is the mean of the predicted probabilities on a training set), so a positive sign means that the feature value contributes to increase the predicted probability of class 1 with respect to the average prediction and a negative sign means that it contributes to reduce the model's output with respect to the average.

Therefore, the individual contributions of features are not comparable between explanation methods. Instead, we have to compare the qualitative understanding between the variables and the class that they provide in order to compare the methods. And, in that sense, note that in all cases LIME and IME provide the same qualitative explanation of the reasons behind each prediction made by the model: for observations 1 and 3 the relevant features are x_1 and x_2 with similar influence on the prediction (i.e., similar absolute value of their contributions), and for observations 2 and 4 the relevant features are x_1 and x_2 with opposite contributions to the model's decision. The same is not true for explanation vectors, which correctly highlights relevant features but, as we discussed in the previous section, its explanations seems to be a little simplistic.

In conclusion, the three methods are different in nature, so they obviously produce different explanations for the same predictions made by the same model. The discussed examples have shown that LIME and IME agree on the reasons behind a model's prediction. Explanation vectors also produce qualitative understanding and it usually mark as most influential the same variables as LIME and IME but, in the analyzed examples, it seems to sometimes provide too simplistic explanations.

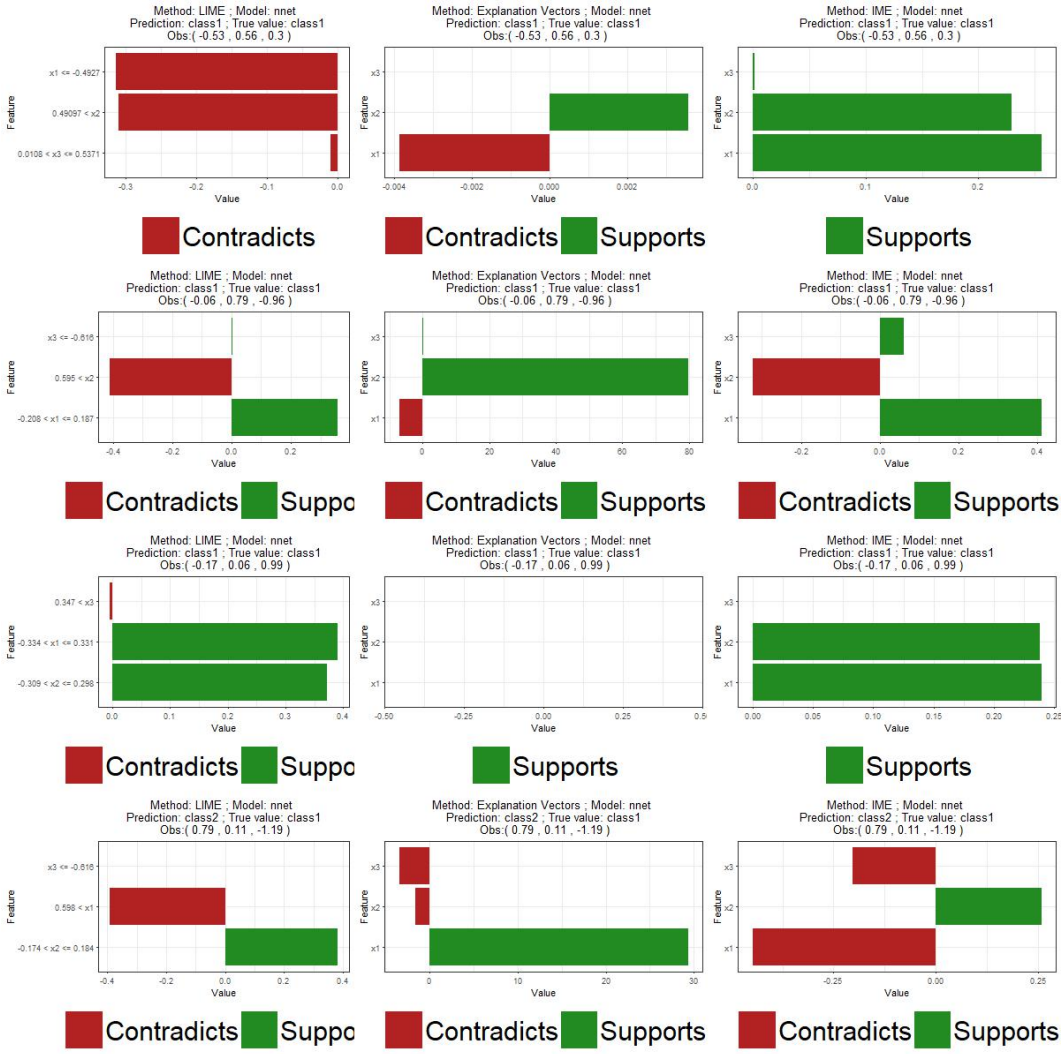


FIG. 7. LIME, explanation vectors and IME explanations for the predictions produced by a neural network for four observations. Each column displays the explanations made by one of the explanation methods: LIME (left), explanation vectors (middle) and IME (right). Each row displays the explanations for one particular instance: $\tilde{x}^{(1)} = (-0.53, 0.56, 0.3)$ (first row), $\tilde{x}^{(2)} = (-0.06, 0.79, -0.96)$ (second row), $\tilde{x}^{(3)} = (-0.17, 0.06, 0.99)$ (third row) and $\tilde{x}^{(4)} = (0.79, 0.11, -1.19)$ (fourth row).

5. Comparing models

Models are approximations to the underlying function that determines the relationship between the features and the response. Different models produce different predictions because the approximation of the true underlying function that they have learned is different. Therefore, the reasons behind the prediction for the same observation of different models can be very different, even if they produce the same prediction. In this section, we will compare the explanations obtained by explanation methods for one observation made by four different models: a logistic regression (glm), a random forest (rf), an artifficial neural network (nnet) and an RBF kernel support vector machine (svmRadial). A logistic regression is clearly not a good choice for this non-linear data set, but we want to see if the explanation

methods are able to notice that. A random forest, a neural network and an RBF kernel SVM are considered because: first, they are popular machine learning techniques with a well-known ability to perform non-linear classification and, second, they are three typical examples of models considered black boxes.

The followed methodology is the one described in section 2. For logistic regression, the data were centered and scaled before training. For the random forest, the number of trees is set to 2,000 and the number of features chosen at each split is optimized using the out-of-bag error. For the neural network, we just considered one hidden layer and we optimized the number of neurons and the decay parameter using 5-fold cross validation. For the RBF kernel SVM, we optimized the regularization parameter C and the gamma parameter γ. The final models obtained are:

(1) Logistic regression with coefficients $\beta_0 = -0.040014017$, $\beta_1 = -0.003815408$, $\beta_2 = -0.017720979$ and $\beta_3 = 0.024192600$. This model has an accuracy of 47% on the test data.
(2) Random forest. The number of trees used is 2,000 and the optimal number of features chosen at each split is 3. This model has an out-of-bag error of 3.2% (i.e., accuracy of 96.8%).
(3) Neural network with a single hidden layer of 5 neurons and decay parameter $\lambda = 10^{-5}$. This model has an accuracy of 98.8% on the test data.
(4) RBF support vector machine with regularization parameter $C = 10$ and gamma parameter $\gamma = 10^{-4}$. This model has an accuracy of 81.4% on the test data.

Figure 8 show the explanations of the predictions for $\tilde{x}^{(2)} = (-0.06, 0.79, -0.96)$ made by the previous four models. On the one hand, we can see that all explanation methods agree on the fact that logistic regression is the only model where the most influential variable is x_3, which we know should not be the case. Therefore, this model correctly predicts the class of the observation but the reasons behind this decision are clearly wrong.

On the other hand, we notice that we reach the same conclusion regardless the explanation method used: the other three models produce their respective predictions due to the same reasons. The clearest example is seen when we use LIME as explanation method: the explanations for the random forest, the neural network and the support vector machine are basically the same, that is, all contributions have both the same sign and approximately the same absolute value. The absolute values differ in the case of the explanation vectors, but the most influential feature is the second one in all three models. The last method, IME, assigns the same role to every feature for the three models, but it determines that the absolute value of the contributions change depending on the model, which means that the reasons behind the model's decision are essentially the same but the feature values affect them to a different extent.

Additionally, we discussed in previous sections that these particular reasons behind the predictions made by the random forest, the neural network and the support vector machine are reasonable, so these models properly captures the relationship between the features and the class for that prediction. However, this does not necessarily mean that the models are perfect (in fact, note that the neural network incorrectly predicts class 2 for this observation of actual class 1) but just an indication in that direction. To either conclude if the models are globally correct or at least decide which one is better, we would need to explain more predictions and define a way to assess the trustworthiness of a model based on individual explanations. This is out of the scope of this master thesis, but related work can be found in Reference [**1**], where Ribeiro et al. propose a method that selects a set of representative instances with explanations to address this question.

In conclusion, explanation methods can be useful to compare models by determining whether a model makes a prediction for justified reasons or not. This way, we can evaluate if a model's prediction is correct but for the wrong reasons, such as the logistic regression, or if it makes a justified mistake, such as the neural network. Furthermore, it has the potential to be the complement to accuracy metrics in the model selection process if we could effectively quantify the trustworthiness of a model

using individual explanations with methods like the suggested by Ribeiro et al., but this is left to future work.

FIG. 8. LIME, explanation vectors and IME explanations for the predictions produced for the same observation $\tilde{x}^{(2)} = (-0.06, 0.79, -0.96)$ by four different models. Each column displays the explanations made by one of the explanation methods: LIME (left), explanation vectors (middle) and IME (right). Each row displays the explanations for one particular model: logistic regression (first row), random forest (second row), neural network (third row) and RBF kernel SVM (fourth row).

Conclusions

In this thesis, we have argued that understanding the reasons behind a model's prediction for a particular instance is essential to trust its decisions. In that sense, we have studied in depth three methods found in the literature: Local Interpretable Model-agnostic Explanations (LIME), explanation vectors and Interactions-based Method for Explanation (IME). These methods are able to explain individual predictions of any model without making any assumption of it, so they bring transparency to machine learning algorithms. In addition, we have unified their notation and implemented functions in R to apply them.

First of all, LIME generates an explanation for a prediction from the components of an interpretable model (for instance, the coefficients in a linear regression) that locally approximates the black box model and which is trained over a new data representation, often binary. We have discussed that this method is able to produce simple explanations even in high dimensional problems because we can decide the maximum number of features used in our explanations. However, the flexibility of LIME has the drawback that there are several hyper parameters that need to be set. Second of all, explanation vectors are defined as gradient vectors of the probability function that characterize how a data point has to be moved to change its prediction. The strengths of this method is that it is efficient because the gradient is usually easy to compute and the explanations are easy to interpret, but our experiment has shown that the generated explanations are usually too simplistic. Finally, IME is based on the idea that the input values of an observation work together to cause a change in the model's prediction with respect to the model's expected output, so this method divides this total change in prediction among the features in a way that is "fair" to their contributions across all possible subsets of features. To do so, IME takes advantage of the game theory, which results in some desirable properties for the explanations, but it also has the inconvenient that it is unfeasible to compute the exact explanations. Nevertheless, the latter is a minor issue because there is a sampling procedure to approximate the explanations.

Moreover, our experiment of chapter 5 has let us answer most of the questions that we had at the beginning of this project. First, we have been able to check that the three analyzed methods produce different explanations to explain model's predictions for different instances. Second, we have seen that their explanations really identify the most influential features in a prediction and give zero credit to irrelevant features when a model has correctly captured the relationship between the explanatory variables and the response. Third, although the individual contributions of each explanatory variable produced by LIME, the explanation vectors and IME cannot be compared due to the difference in nature of these methods, we have seen that the qualitative interpretation of the explanations generated by LIME and IME essentially agree on the reasons behind a model's prediction, which is a remarkable result. Finally, we have verified that the explanation methods can also be useful to compare models by determining whether a model makes a prediction for justified reasons or not.

Nevertheless, this work is just an introduction to the study of model-independent explanation methods, and there are several avenues of future work that could be explored. One of them is the application of the studied methods to a regression task. Another could be the study of different fidelity functions, explanation families and weight functions in LIME. Finally, research on methods to assess the global trustworthiness of a model based on individual explanations.

References

[1] MT. Ribeiro, S. Singh, C. Guestrin. "Why should I trust you?" Explaining the Predictions of Any Classifier. In Knowledge Discovery and Data Mining (KDD), 2016.

[2] MT. Ribeiro, S. Singh, C. Guestrin. Model-Agnostic Interpretability of Machine Learning. In Human Interpretability in Machine Learning workshop, ICML '16, 2016.

[3] MT. Ribeiro, S. Singh, C. Guestrin. Nothing Else Matters: Model-Agnostic Explanations By Identifying Prediction Invariance. In NIPS 2016 Workshop on Interpretable Machine Learning in Complex Systems, 2016.

[4] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, KR. MÃ$\frac{1}{4}$ller. How to Explain Individual Classification Decisions. *Journal of Machine Learning Research*, 11, 2010.

[5] M. Robnik-Sikonja, I. Kononenko. Explaining Classifications For Individual Instances. *IEEE Transactions on Knowledge and Data Engineering*, 20(5), 2008.

[6] E. Štrumbelj, I. Kononenko, M. Robnik Sikonja. Explaining instance classifications with interactions of subsets of feature values. *Data & Knowledge Engineering*, 68(10), 2009.

[7] E. Štrumbelj, I. Kononenko. An Efficient Explanation of Individual Classifications using Game Theory. *Journal of Machine Learning Research*, 11, 2010.

[8] E. Štrumbelj, I. Kononenko. A General Method for Visualizing and Explaining Black-Box Regression. In: Dobnikar A, Lotric U, Ster B (eds) ICANNGA (2), vol 6594 of Lecture notes in computer science. Springer, Berlin, pp 21-30, 2011.

[9] SM. Lundberg, SI. Lee. A Unified Approach to Interpreting Model Predictions. arXiv preprint arXiv:1705.07874, 2017.

[10] MT. Ribeiro. Lime: Explaining the predictions of any machine learning classifier. Github Repository https://github.com/marcotcr/lime , 2016.

[11] T.L. Pedersen, M. Benesty. 'lime'. R package, 2017.

[12] M. Robnik-Sikonja. 'ExplainPrediction'. R package, 2017.

[13] Kuhn, M. 'caret' package. Journal of Statistical Software, 28(5), 2008.

[14] Wickham, H. 'ggplot2'. R package, 2009.

Appendix A
R code

Explanation methods

```
explain <- function(model, X_tilde, method="all", predict, type, class,
                    X_train, N, X_test, sigma.v,
                    K, sigma, bin_continuous, n_bins = 4, quantile_bins = TRUE,
                    labels = NULL, n_labels = 1, feature_select = "lasso_path",
                    dist_fun = "euclidean"){
  p <- ncol(X_tilde)
  n <- nrow(X_train)
  if(type=="class"){
    f <- function(x){ifelse(eval(parse(text=predict))==class, 1, 0)}
  }else{
    f <- function(x){
      if(is.vector(x)){x <- setNames(data.frame(t(x)), names(X_tilde))}
      return(eval(parse(text=predict)))
    }
  }
  if(method=="all"|method=="lime"){
    if(length(K)>1|length(sigma)>1|length(bin_continuous)>1|length(n_bins)>1){
      params <- do.call(rbind.data.frame,
                        apply(X_tilde, 1,
                              function(x){
                                x <- setNames(data.frame(t(x)), names(X_tilde))
                                lime_optimizehyperparams(model=model, x_tilde=x,
                                        X_train=X_train, K.v=K, sigma.v=sigma,
                                        bin_continuous.v=bin_continuous, n_bins.v=n_bins, N=N,
                                        quantile_bins=quantile_bins, labels=labels,
                                        n_labels=n_labels, feature_select=feature_select,
                                        dist_fun=dist_fun)}
                        )
      )
      explanation.lime <- do.call(rbind.data.frame,
                                  lapply(1:nrow(X_tilde),
                                    function(i){
                                      lime(model, X_tilde[i,], X_train, N=N,
                                           K=params$K[i], sigma=params$sigma[i],
                                           bin_continuous=params$bin_continuous[i],
                                           n_bins=params$n_bins[i],
```

```
                                                    quantile_bins=quantile_bins, labels=labels,
                                                    n_labels=n_labels,
                                                    feature_select=feature_select,
                                                    dist_fun=dist_fun)
                                                    }))
      explanation.lime <- reshape(explanation.lime[,-c(8:9,12:13)],
                                  idvar = names(explanation.lime)[1:7],
                                  timevar = "feature_desc", direction = "wide")
      explanation.lime <- list(explanation=explanation.lime[,c(-(1:5),-7)],
                               parameters=params, label=explanation.lime$label,
                               r2=explanation.lime$model_r2)
      }else{
      explanation.lime <- lime(model, X_tilde, X_train, K, sigma, N,
                              bin_continuous, n_bins, quantile_bins,
                              labels, n_labels, feature_select,
                              dist_fun)
      explanation.lime <- reshape(explanation.lime[,-c(8:9,12:13)],
                                  idvar = names(explanation.lime)[1:7],
                                  timevar = "feature_desc", direction = "wide")
      explanation.lime <- list(explanation=explanation.lime[,c(-(1:5),-7)],
                               label=explanation.lime$label, r2=explanation.lime$model_r2)
    }
    row.names(explanation.lime$explanation) <- c()
    names(explanation.lime$explanation)[-1] <- substr(names(explanation.lime$explanation)[-1],
                                                      16,
                                                      nchar(names(explanation.lime$explanation)[-1]))
    if(method=="all"){
      return(list(lime=explanation.lime,
                  ev=ev(model, f, type, class, X_tilde, X_train, X_test, sigma.v, n, p),
                  ime=ime(model, f, type, class, X_tilde, X_train, N, n, p)))
    }
    if(method=="lime"){
      return(explanation.lime)
    }
  }
  if(method=="ev"){
    return(ev(model, f, type, class, X_tilde, X_train, X_test, sigma.v, n, p))
  }
  if(method=="ime"){
    return(ime(model, f, type, class, X_tilde, X_train, N, n, p))
  }
}

#### IME ####
ime <- function(model, f, type, class, X_tilde, X_train, N, n, p){
  orderinstance <- function(i, n, p, X_tilde, X_train){
    perm <- sample(1:p, p)
    z <- X_train[sample(1:n, 1),]
    pos_i <- which(perm==i)
    X1 <- setNames(data.frame(tcrossprod(rep(1,nrow(X_tilde)),
                  as.numeric(z))), names(X_tilde))
```

```
    if(pos_i>1){
      X1[,perm[1:(pos_i-1)]] <- X_tilde[,perm[1:(pos_i-1)]]
    }
    X2 <- X1
    X1[,i] <- X_tilde[,i]
    return(f(X1)-f(X2))
  }
  phi <- lapply(1:p,
                function(i){
                  phi_i <- replicate(N, orderinstance(i=i, n=n, p=p,
                                                      X_tilde=X_tilde, X_train=X_train))
                  if(is.null(dim(phi_i))){
                    return(mean(phi_i))
                  }else{
                    return(apply(phi_i, 1, mean))
                  }
                })
  return(setNames(data.frame(phi), names(X_train)))
}


#### EXPLANATION VECTORS ####
ev <- function(model, f, type, class, X_tilde, X_train, X_test, sigma.v, n, p){
  library(numDeriv)
  if(type=="prob"){
    ev <- apply(X_tilde, 1, function(x_tilde){grad(function(x){1-f(x)}, x_tilde)})
    return(setNames(data.frame(t(ev)), names(X_tilde)))
  }
  if(type=="class"){
    y_train <- f(X_train)
    y_test <- f(X_test)
    k_sigma <- function(Z, s){ exp(-0.5*diag(Z\%*\%t(Z))/s^2)/sqrt(2*pi*s^2) }
    kde <- function(x, sigma, I_c){
      Z <- matrix(rep(as.numeric(x),n), ncol=p, byrow=T) - as.matrix(X_train)
      k_i <- k_sigma(Z, s=sigma)
      fhat <- sum(k_i[I_c])/sum(k_i)
      return(fhat)
    }
    fhat <- function(X, sigma){
      I_c <- which(y_train==1)
      if(is.null(dim(X))){
        fhat <- kde(X, sigma=sigma, I_c=I_c)
      }else{
        fhat <- apply(X, 1, kde, sigma=sigma, I_c=I_c)
      }
      return(fhat)
    }
    sigma.acc <- sapply(sigma.v,
                        function(sigma){
                          yhat_test <- ifelse(fhat(X_test, sigma=sigma)>0.5, 1, 0)
                          sum(y_test==yhat_test, na.rm=T)
```

```
                        }
    )
    sigma <- sigma.v[which.max(sigma.acc)]
    ev <- apply(X_tilde, 1, function(x_tilde){grad(function(x, sigma){1-fhat(x, sigma)},
                as.numeric(x_tilde), sigma=sigma)})
    return(list(ev=setNames(data.frame(t(ev)), names(X_tilde)),
                sigma.acc=sigma.acc,
                sigma=sigma,
                f_hat=fhat(X_test, sigma=sigma)))
  }
}

#### LIME ####
lime <- function(model, X_tilde, X_train, K, sigma, N = 5000,
                 bin_continuous=FALSE, n_bins = 4, quantile_bins = TRUE,
                 labels = NULL, n_labels = 1, feature_select = "lasso_path",
                 dist_fun = "euclidean"){
  library(lime)
  explainer <- lime::lime(x=X_train, model=model, bin_continuous=bin_continuous,
                          n_bins = n_bins, quantile_bins = quantile_bins)
  explanation <- NA
  attempt <- 1
  while( is.na(explanation) && attempt <= 3 ) {
    attempt <- attempt + 1
    try(
      explanation <- tryCatch({lime::explain(x=X_tilde, explainer=explainer, labels = labels,
                                             n_labels = n_labels,  n_features=K,
                                             n_permutations = N,
                                             feature_select = feature_select,
                                             dist_fun = dist_fun,
                                             kernel_width = sigma)},
                              error=function(e){
                                return(NA)}
      )
    )
  }
  return(explanation)
}

lime_optimizehyperparams <- function(model, x_tilde, X_train, K.v, sigma.v,
                                     bin_continuous.v, n_bins.v, N,  quantile_bins,
                                      labels, n_labels, feature_select, dist_fun){
  params_grid <- expand.grid(K=K.v, sigma=sigma.v, bin_continuous=bin_continuous.v,
                             n_bins=n_bins.v, KEEP.OUT.ATTRS=FALSE)
  params_grid$r2 <- apply(params_grid, 1,
                          function(param){
                            r2 <- NA
                            attempt <- 1
                            while( is.na(r2) && attempt <= 3 ) {
                              attempt <- attempt + 1
                              try(
```

```
                                r2 <- tryCatch({explanation <- lime(model, x_tilde, X_train,
                                                K=param[1], sigma=param[2], N=N,
                                                bin_continuous=param[3],
                                                n_bins=param[4],
                                                quantile_bins=quantile_bins,
                                                labels=labels, n_labels=n_labels,
                                                feature_select=feature_select,
                                                dist_fun=dist_fun)
                                reshape(explanation[,-c(8:9,12:13)],
                                        idvar = names(explanation)[1:7],
                                        timevar = "feature_desc",
                                        direction = "wide")$model_r2},
                                error=function(e){
                                  return(NA)}
                                )
                              )
                            }
                            return(r2)
                          })
  #print(params_grid)
  return(params_grid[which.max(params_grid$r2),])
}
```

Plot explanations

```
plot_explanations <- function(explanation, x_tilde, y_tilde, model, method){
        df <- setNames(data.frame(t(explanation[,!is.na(explanation)])), "Value")
        if(nrow(df)==1){row.names(df) <- names(explanation)[!is.na(explanation)]}
        df$sign <- factor(ifelse(df$Value<0, "Contradicts", "Supports"))
        df$Feature <- row.names(df)
        if(all(df$Value>=0)){
                ggplot(data=df, aes(x=Feature, y=Value, fill=sign)) +
                        geom_bar(stat="identity") + coord_flip() +
                        scale_fill_manual(values=c("forestgreen")) +
                        ggtitle(paste("Method:",method,"; Model:",model$method,
                                "\n Prediction:", predict(model,x_tilde),"; True value:",
                                y_tilde,"\n Obs:(",
                        paste(as.character(round(x_tilde,2)), collapse=', '),")")) +
                        theme_bw() +
                        theme(legend.title=element_blank(),
                                plot.title = element_text(hjust = 0.5),
                                legend.position="bottom",
                                legend.text=element_text(size=30),
                                legend.key.size = unit(3,"line"))
        }else{
                ggplot(data=df, aes(x=Feature, y=Value, fill=sign)) +
                        geom_bar(stat="identity") + coord_flip() +
                        scale_fill_manual(values=c("firebrick", "forestgreen")) +
                        ggtitle(paste("Method:",method,"; Model:",model$method,
                                "\n Prediction:", predict(model,x_tilde),"; True value:",
                                y_tilde,"\n Obs:(",
                        paste(as.character(round(x_tilde,2)), collapse=', '),")")) +
```

```
                        theme_bw() +
                        theme(legend.title=element_blank(),
                                plot.title = element_text(hjust = 0.5),
                                legend.position="bottom",
                                legend.text=element_text(size=30),
                                legend.key.size = unit(3,"line"))
        }
}

# Multiple plot function
# Source: http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

  if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}
```

Main code

```
#example data
set.seed(0)
library(plotly)
```

```
cylinder.data <- setNames(data.frame(mlbench.circle(10000,2)), c("x1","x2","class"))
cylinder.data$class <- ifelse(cylinder.data$class==1, "class1", "class2")
cylinder.data <- cbind(cylinder.data, data.frame(x3=runif(10000,-1,1)))
plot_ly(cylinder.data, x = ~x1, y = ~x2, z = ~x3, color = ~class,
  colors = c('#BF382A', '#0C4B8E')) %>%
  add_markers() %>%
 layout(scene = list(xaxis = list(title = 'x1'),
                     yaxis = list(title = 'x2'),
                     zaxis = list(title = 'x3')))

set.seed(123)
#data
library(mlbench)
cylinder.train <- setNames(data.frame(mlbench.circle(1000,2)), c("x1","x2","class"))
cylinder.train$class <- ifelse(cylinder.train$class==1, "class1", "class2")
cylinder.test <- setNames(data.frame(mlbench.circle(500,2)), c("x1","x2","class"))
cylinder.test$class <- ifelse(cylinder.test$class==1, "class1", "class2")
cylinder.train <- cbind(cylinder.train, data.frame(x3=runif(1000,-1,1)))
cylinder.test <- cbind(cylinder.test, data.frame(x3=rnorm(500,-1,1)))
library(GGally)
library(ggplot2)
ggsave(filename="documents/LaTeX/images/cylinder_train.jpg",
       plot=ggpairs(cbind(cylinder.train[,-3],
       data.frame(class=cylinder.train$class)), aes(colour = class, alpha = 0.4)) +
       theme_bw())
plot(cylinder.test[,1:2],col=factor(cylinder.test[,3]),pch=16,cex=0.75)
text(x=cylinder.test[,1],y=cylinder.test[,2]+0.05,labels=row.names(cylinder.test),
cex=0.75)

#models
library(caret)
cylinder.glm <- train(class ~ ., data=cylinder.train,
                preProcess=c('center','scale'),
                    trControl=trainControl(method="cv", number=5),
                    method="glm", family="binomial")
cylinder.rf <- train(class ~ ., data=cylinder.train, method="rf", ntree=2000,
                   trControl=trainControl(method="oob", classProbs=TRUE),
                   tuneGrid=data.frame(mtry=1:3))
cylinder.nnet <- train(class ~ ., data=cylinder.train, method="nnet",
                     trControl=trainControl(method="cv", number=5), maxit = 1000,
                     trace = F, tuneGrid=expand.grid(size=seq(1,15,by=2),
                     decay=10^seq(-5,3,by=2)))
cylinder.svm <- train(class ~ ., data=cylinder.train, method="svmRadial",
                    trControl=trainControl(method="cv", number=5, classProbs=TRUE),
                    tuneGrid=expand.grid(C=10^seq(-3,3,by=1),
                    sigma=10^seq(-4,4,by=1)))

sort(coef(cylinder.glm$finalModel))
cylinder.rf$bestTune
cylinder.nnet$bestTune
cylinder.svm$bestTune
```

```
confusionMatrix(predict(cylinder.glm, cylinder.test), cylinder.test$class)
confusionMatrix(predict(cylinder.rf, cylinder.test), cylinder.test$class)
confusionMatrix(predict(cylinder.nnet, cylinder.test), cylinder.test$class)
confusionMatrix(predict(cylinder.svm, cylinder.test), cylinder.test$class)

#plot gradients
X_tilde <- cylinder.test[c(481,137,29,302),-3]
Y_tilde <- cylinder.test[c(481,137,29,302),3]

plot_gradient <- function(model, X_tilde, bins=10){
  grid.xy <- expand.grid(x1=seq(-1,1,by=0.01),x2=seq(-1,1,by=0.01),x3=0)
  grid.xy$density <- predict(model, grid.xy, type="prob")[,1]
  ggplot(grid.xy, aes(x=x1, y=x2)) +
    geom_raster(data=grid.xy, aes(fill=density)) +
    geom_contour(data=grid.xy, aes(x=x1, y=x2, z=density), bins=bins) +
    scale_colour_gradient(guide = 'none') +
    geom_point(data=cylinder.train[cylinder.train$class=="class1",],
    aes(x=x1,y=x2), fill="palevioletred", size=2, pch=21) +
    geom_point(data=cylinder.train[cylinder.train$class=="class2",],
    aes(x=x1,y=x2), fill="paleturquoise", size=2, pch=21) +
    geom_point(data=X_tilde, aes(x=x1,y=x2), colour="palevioletred", size=8, pch=17) +
    geom_text(data=X_tilde, aes(label=as.character(1:nrow(X_tilde))), hjust=-0.5,
    vjust=0, size = 10, colour="white") +
    ggtitle(paste("Model:",model)) +
    theme_bw() + theme(plot.title = element_text(hjust = 0.5))
}

multiplot(plot_gradient(cylinder.glm, X_tilde),
        plot_gradient(cylinder.rf, X_tilde),
          plot_gradient(cylinder.nnet, X_tilde),
          plot_gradient(cylinder.svm, X_tilde), cols=2)

#explanations
cylinder.glm.explanations <- explain(model=cylinder.glm, X_tilde=X_tilde,
                             X_train=cylinder.train[,-3], method="all",
                                    predict="predict(model, x, type='prob')[,1]",
                                    type="prob",
                                    N=10000, K=3, sigma=10^seq(-5,5,by=1),
                                    bin_continuous=TRUE, n_bins=2:5)
cylinder.rf.explanations <- explain(model=cylinder.rf, X_tilde=X_tilde,
                            X_train=cylinder.train[,-3], method="all",
                                   predict="predict(model, x)", type="class",
                                   class="class1", X_test=cylinder.test[,-3],
                                   sigma.v=10^seq(-1,1,by=1),
                                   N=10000, K=3, sigma=10^seq(-5,5,by=1),
                                   bin_continuous=TRUE, n_bins=2:5)
cylinder.nnet.explanations <- explain(model=cylinder.nnet, X_tilde=X_tilde,
                               X_train=cylinder.train[,-3], method="all",
                                      predict="predict(model, x, type='prob')[,1]",
                                      type="prob",
```

```
                                    N=10000, K=3, sigma=10^seq(-5,5,by=1),
                                    bin_continuous=TRUE, n_bins=2:5)
cylinder.svm.explanations <- explain(model=cylinder.svm, X_tilde=X_tilde,
                                X_train=cylinder.train[,-3], method="all",
                                     predict="predict(model, x, type='prob')[,1]",
                                     type="prob",
                                    N=10000, K=3, sigma=10^seq(-5,5,by=1),
                                    bin_continuous=TRUE, n_bins=2:5)

#explaining predictions (nnet)
ggsave(filename="documents/LaTeX/images/cylinder_density_nnet.jpg",
       plot=plot_gradient(cylinder.nnet, X_tilde))
ggsave(filename="outputs/cylinder_density_nnet.jpg",
       plot=plot_gradient(cylinder.nnet, X_tilde))

#lime
nnet.lime <- as.list(rep(NULL, nrow(X_tilde)))
for(i in 1:nrow(X_tilde)){
  x_tilde <- X_tilde[i,]
  y_tilde <- Y_tilde[i]
  if(cylinder.nnet.explanations$lime$label[i]=="class1"){
    nnet.lime[[i]] <- plot_explanations(
                                cylinder.nnet.explanations$lime$explanation[i,-1],
                                x_tilde, y_tilde, cylinder.nnet, method="LIME")
  }else{
    nnet.lime[[i]] <- plot_explanations(
                                -cylinder.nnet.explanations$lime$explanation[i,-1],
                                x_tilde, y_tilde, cylinder.nnet, method="LIME")
  }
}
jpeg(filename = "outputs/cylinder_nnet_lime.jpg", width=1200, height=1000)
multiplot(nnet.lime[[1]], nnet.lime[[3]],
          nnet.lime[[2]], nnet.lime[[4]], cols=2)
dev.off()

#ev
nnet.ev <- as.list(rep(NULL, nrow(X_tilde)))
for(i in 1:nrow(X_tilde)){
  x_tilde <- X_tilde[i,]
  y_tilde <- Y_tilde[i]
  nnet.ev[[i]] <- plot_explanations(-cylinder.nnet.explanations$ev[i,],
                        x_tilde, y_tilde, cylinder.nnet,
                        method="Explanation_Vectors")
}
jpeg(filename = "outputs/cylinder_nnet_ev.jpg", width=1200, height=1000)
multiplot(nnet.ev[[1]], nnet.ev[[3]],
          nnet.ev[[2]], nnet.ev[[4]], cols=2)
dev.off()

#ime
nnet.ime <- as.list(rep(NULL, nrow(X_tilde)))
```

```
for(i in 1:nrow(X_tilde)){
  x_tilde <- X_tilde[i,]
  y_tilde <- Y_tilde[i]
  nnet.ime[[i]] <- plot_explanations(cylinder.nnet.explanations$ime[i,],
                          x_tilde, y_tilde, cylinder.nnet, method="IME")
}
jpeg(filename = "outputs/cylinder_nnet_ime.jpg", width=1200, height=1000)
multiplot(nnet.ime[[1]], nnet.ime[[3]],
          nnet.ime[[2]], nnet.ime[[4]], cols=2)
dev.off()

#comparing methods
jpeg(filename = "outputs/cylinder_nnet_explanations_obs481.jpg",
        width=1200, height=400)
multiplot(nnet.lime[[1]], nnet.ev[[1]], nnet.ime[[1]], cols=3)
dev.off()

nnet.lime <- as.list(rep(NULL, nrow(X_tilde)))
nnet.ev <- as.list(rep(NULL, nrow(X_tilde)))
nnet.ime <- as.list(rep(NULL, nrow(X_tilde)))
for(i in 1:nrow(X_tilde)){
  x_tilde <- X_tilde[i,]
  y_tilde <- Y_tilde[i]
  if(cylinder.nnet.explanations$lime$label[i]=="class1"){
    nnet.lime[[i]] <- plot_explanations(
                                cylinder.nnet.explanations$lime$explanation[i,-1],
                                x_tilde, y_tilde, cylinder.nnet, method="LIME")
  }else{
    nnet.lime[[i]] <- plot_explanations(
                               -cylinder.nnet.explanations$lime$explanation[i,-1],
                                x_tilde, y_tilde, cylinder.nnet, method="LIME")
  }
  nnet.ev[[i]] <- plot_explanations(
                          cylinder.nnet.explanations$ev[i,],
                          x_tilde, y_tilde, cylinder.nnet,
                          method="Explanation_Vectors")
  nnet.ime[[i]] <- plot_explanations(
                          cylinder.nnet.explanations$ime[i,], x_tilde, y_tilde,
                          cylinder.nnet, method="IME")
}

jpeg(filename = "outputs/cylinder_nnet_comparing_methods.jpg",
        width=1200, height=1200)
multiplot(nnet.lime[[1]], nnet.lime[[2]], nnet.lime[[3]], nnet.lime[[4]],
          nnet.ev[[1]], nnet.ev[[2]], nnet.ev[[3]], nnet.ev[[4]],
          nnet.ime[[1]], nnet.ime[[2]], nnet.ime[[3]], nnet.ime[[4]], cols=3)
dev.off()

#comparing models
for(i in 1:nrow(X_tilde)){
  x_tilde <- X_tilde[i,]
```

```
y_tilde <- Y_tilde[i]
if(cylinder.glm.explanations$lime$label[i]=="class1"){
  glm.lime <- plot_explanations(cylinder.glm.explanations$lime$explanation[i,-1],
                        x_tilde, y_tilde, cylinder.glm, method="LIME")
}else{
  glm.lime <- plot_explanations(-cylinder.glm.explanations$lime$explanation[i,-1],
                        x_tilde, y_tilde, cylinder.glm, method="LIME")
}
glm.ev <- plot_explanations(cylinder.glm.explanations$ev[i,], x_tilde, y_tilde,
                        cylinder.glm, method="Explanation_Vectors")
glm.ime <- plot_explanations(cylinder.glm.explanations$ime[i,], x_tilde, y_tilde,
                        cylinder.glm, method="IME")
if(cylinder.rf.explanations$lime$label[i]=="class1"){
  rf.lime <- plot_explanations(cylinder.rf.explanations$lime$explanation[i,-1],
                        x_tilde, y_tilde, cylinder.rf, method="LIME")
}else{
  rf.lime <- plot_explanations(-cylinder.rf.explanations$lime$explanation[i,-1],
                        x_tilde, y_tilde, cylinder.rf, method="LIME")
}
rf.ev <- plot_explanations(cylinder.rf.explanations$ev$ev[i,], x_tilde, y_tilde,
                        cylinder.rf, method="Explanation_Vectors")
rf.ime <- plot_explanations(cylinder.rf.explanations$ime[i,], x_tilde, y_tilde,
                        cylinder.rf, method="IME")
if(cylinder.nnet.explanations$lime$label[i]=="class1"){
  nnet.lime <- plot_explanations(
                        cylinder.nnet.explanations$lime$explanation[i,-1],
                         x_tilde, y_tilde, cylinder.nnet, method="LIME")
}else{
  nnet.lime <- plot_explanations(
                        -cylinder.nnet.explanations$lime$explanation[i,-1],
                        x_tilde, y_tilde, cylinder.nnet, method="LIME")
}
nnet.ev <- plot_explanations(cylinder.nnet.explanations$ev[i,], x_tilde, y_tilde,
                        cylinder.nnet, method="Explanation_Vectors")
nnet.ime <- plot_explanations(cylinder.nnet.explanations$ime[i,],
                        x_tilde, y_tilde, cylinder.nnet, method="IME")
if(cylinder.svm.explanations$lime$label[i]=="class1"){
  svm.lime <- plot_explanations(
                        cylinder.svm.explanations$lime$explanation[i,-1], x_tilde,
                        y_tilde, cylinder.svm, method="LIME")
}else{
  svm.lime <- plot_explanations(
                        -cylinder.svm.explanations$lime$explanation[i,-1], x_tilde,
                        y_tilde, cylinder.svm, method="LIME")
}
svm.ev <- plot_explanations(cylinder.svm.explanations$ev[i,], x_tilde, y_tilde,
                        cylinder.svm, method="Explanation_Vectors")
svm.ime <- plot_explanations(cylinder.svm.explanations$ime[i,], x_tilde, y_tilde,
                        cylinder.svm, method="IME")
jpeg(filename = paste0("outputs/cylinder_explanations_",i,"_obs",
      row.names(x_tilde),".jpg"), width=1200, height=1000)
```

```
  multiplot(glm.lime, rf.lime, nnet.lime, svm.lime,
            glm.ev, rf.ev, nnet.ev, svm.ev,
            glm.ime, rf.ime, nnet.ime, svm.ime,
            cols=3)
  dev.off()
}
}
```